Multimodal Transformers for Biomedical Text and Knowledge Graph Data

Helena Balabin

Publisher: Dean Prof. Dr. Wolfgang Heiden

**Hochschule Bonn-Rhein-Sieg – University of Applied Sciences,
Department of Computer Science**

Sankt Augustin, Germany

January 2022

Technical Report 02-2022

Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

ISSN 1869-5272

ISBN 978-3-96043-100-8

Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Fraunhofer
SCAI

This work was supervised by	Paul G. Plöger Martin Hofmann-Apitius Daniel Domingo-Fernández

Digital Object Identifier https://doi.org/10.18418/978-3-96043-100-8

Abstract

Recent advances in Natural Language Processing have substantially improved contextualized representations of language. However, the inclusion of factual knowledge, particularly in the biomedical domain, remains challenging. Hence, many Language Models (LMs) are extended by Knowledge Graphs (KGs), but most approaches require entity linking (i.e., explicit alignment between text and KG entities). Inspired by single-stream multimodal Transformers operating on text, image and video data, this thesis proposes the Sophisticated Transformer trained on biomedical text and Knowledge Graphs (STonKGs). STonKGs incorporates a novel multimodal architecture based on a cross encoder that uses the attention mechanism on a concatenation of input sequences derived from text and KG triples, respectively. Over 13 million so-called text-triple pairs, coming from PubMed and assembled using the Integrated Network and Dynamical Reasoning Assembler (INDRA), were used in an unsupervised pre-training procedure to learn representations of biomedical knowledge in STonKGs. By comparing STonKGs to an NLP- and a KG-baseline (operating on either text or KG data) on a benchmark consisting of eight fine-tuning tasks, the proposed knowledge integration method applied in STonKGs was empirically validated. Specifically, on tasks with a comparatively small dataset size and a larger number of classes, STonKGs resulted in considerable performance gains, beating the F1-score of the best baseline by up to 0.083. Both the source code as well as the code used to implement STonKGs are made publicly available so that the proposed method of this thesis can be extended to many other biomedical applications.

Contents

List of Algorithms

List of Figures

List of Tables

List of Abbreviations

Aβ amyloid-β.

AD alzheimer's disease.

AdamW adaptive moment estimation with decoupled weight decay.

API application programming interface.

BACE beta-secretase.

BEL biological expression language.

BERT bidirectional encoder representations from transformers.

BERT-MK bidirectional encoder representations from transformers based language model with medical knowledge.

BFS breadth first search.

BioKGLM biomedical contextualized language model by incorporating biomedical knowledge graphs.

BioPAX biological pathway exchange.

CBOW continuous bag-of-words.

CDR chemical-disease relation.

ChEBI chemical entities of biological interest.

CLIP contrastive language–image pre-training.

CLO cell line ontology.

CMU-MOSEI carnegie mellon university multimodal opinion sentiment and emotion intensity.

CNN convolutional neural network.

COCO common objects in context.

CoLAKE contextualized language and knowledge embedding.

CTD comparative toxicogenomics database.

dEA denoising entity auto-encoder.

DFS depth first search.

DKRL description-embodied knowledge representation learning.

DL deep learning.

DO disease ontology.

dVAE discrete variational autoencoder.

ELECTRA efficiently learning an encoder that classifies token replacements accurately.

ENT-DESC entity-to-description.

ERNIE enhanced language representation with informative entities.

FB15K freebase 15k.

FP16 half precision floating point format.

GAN generative adversarial network.

GAT graph attention network.

GCN graph convolutional network.

GF graph factorization.

GLUE general language understanding evaluation.

GPT generative pre-trained transformer.

GPU graphics processing unit.

GraphSAGE graph sample and aggregate.

HGNC human genome organisation gene nomenclature committee.

HMS harvard medical school.

HPO hyperparameter optimization.

HTML hypertext markup language.

INDRA integrated network and dynamical reasoning assembler.

ITM image-text-matching.

JSON javascript object notation.

JSONL javascript object notation lines.

K-BERT knowledge-enabled bidirectional encoder representations from transformers.

KEPLER knowledge embedding and pre-trained language representation.

KG knowledge graph.

KGC knowledge graph completion.

KGE knowledge graph embedding.

KGEM knowledge graph embedding model.

KGPT knowledge-grounded pre-training.

KRL knowledge representation learning.

LM language model.

LSA latent semantic analysis.

LSTM long short-term memory.

LSTRC literature selection technical review committee.

LXMERT learning cross-modality encoder representations from transformers.

MDETR modulated detection transformer.

MEM masked entity modeling.

MeSH medical subject headings.

ML machine learning.

MLM masked language modeling.

MRI magnetic resonance imaging.

MRM masked region modeling.

NCBI (U.S.) national center for biotechnology information.

NER named entity recognition.

NIH (U.S.) national institutes of health.

NLM (U.S.) national library of medicine.

NLP natural language processing.

NLU natural language understanding.

NRL network representation learning.

NSP next sentence prediction.

ODE ordinary differential equation.

OOD out of distribution.

OOV out-of-vocabulary.

PCA principal component analysis.

PMC pubmed central.

PPI protein protein interaction.

RAM random-access memory.

RE relation extraction.

REACH reading and assembling contextual and holistic mechanisms from text.

RNN recurrent neural network.

ROI region of interest.

SGD stochastic gradient descent.

SIGNOR signaling network open resource.

SIRT3 sirtuin-3.

SMP symmetric multiprocessing.

SSP semantic space projection.

STonKGs sophisticated transformer trained on biomedical text and knowledge graphs.

SuperGLUE super general language understanding evaluation.

TPU tensor processing unit.

TRIPS the rochester interactive planner system.

UNITER universal image-text representation.

ViLBERT vision and language bidirectional encoder representations from transformers.

VQA visual question answering.

WebNLG web natural language generation.

WK graph word knowledge graph.

1 Introduction

The purpose of the following sections is to outline the rationale and the goals of this thesis. First, a high-level introduction to the subjects covered in this thesis is given in Section 1.1. Next, the investigated hypothesis is stated in Section 1.2. Lastly, the steps required to examine the hypothesis of this thesis are formulated as concrete aims in Section 1.3.

1.1 Context in Biomedicine

In general, the process of drug discovery proves to be a tedious, risky and costly endeavour, as shown in the ongoing COVID-19 crisis [Shi+20]. Although many clinical trials pass the early stages, they are retracted at late stages due to a lack of efficiency or severe unexpected side effects. For instance, multiple promising drugs for Alzheimer's disease (AD) were based on Beta-secretase (BACE) 1 inhibitors, since they proved to be effective in reducing the accumulation of amyloid-β (Aβ) peptides (a major contributing factor in AD) in mice [DY19]. However, this effect has not been successfully replicated in human subjects yet, resulting in costly failures of multiple clinical trials [DY19]. To prevent such failures and accelerate the discovery process, the assessment of each scientific discovery must incorporate contextual information (such as the species in which a given biological interaction is occurring).

Throughout the last years, Machine Learning (ML), specifically Deep Learning (DL), has played a crucial role in accelerating biomedical research (see, e.g., [Mio+18; MLY17] for related surveys). Context is typically integrated through multiple distinct data sources, so-called modalities, in the same ML model. One such example is the ML-based classification of disease stages in AD (see, e.g., [Tan+20] for a review). It is typically based on, inter alia, a combination of cognitive assessments, demographics and Magnetic Resonance Imaging (MRI) features [Bir+20]. However, the notion of context learned by an ML model is highly dependent on the quality and size of the dataset used for training. Especially in the biomedical domain, many ML applications are limited by the small number of participants in the clinical trials used as a basis for training the model.

In contrast, efforts have been made to design large-scale ML models based on vast amounts of biomedical literature, which focus on learning context through the surrounding information of a given input. Such models are typically trained in a transfer learning setting, meaning they are trained on large training datasets in a semi-supervised or unsupervised manner and then transferred to specific tasks by further training the same model on much smaller task-specific datasets. Essentially, there are two ways for learning contextualized

ML models from literature: through structured information in the form of Knowledge Graphs (KGs) learned by Network Representation Learning (NRL) approaches or through unstructured information in the form of text data in Language Models (LMs), learned by Natural Language Processing (NLP) methods. Recent comparisons and surveys on existing NRL methods and LMs in the biomedical domain can be found in, e.g., [Su+20] and [Lew+20], respectively.

Both NRL and LMs share the goal of projecting input features into high-dimensional vector spaces, and these projections are commonly referred to as embeddings. Nonetheless, only a few approaches in the biomedical domain, such as the Bidirectional Encoder Representations from Transformers based language model integrated with Medical Knowledge (BERT-MK) [He+20], focus on utilizing information from both modalities, i.e. both Knowledge Graphs and text corpora. Moreover, these multimodal approaches typically rely on explicit alignments between text and KG entities, which leads to limited use of the available KG data. Given the recent successes of novel large-scale multimodal ML models in the general domain, such as DALL-E [Ram+21] or Contrastive Language–Image Pre-training (CLIP) [Rad+21], there is considerable potential for using methodological advances in the general domain to develop an improved model leveraging both text and KG embeddings in the biomedical field.

1.2 Hypothesis

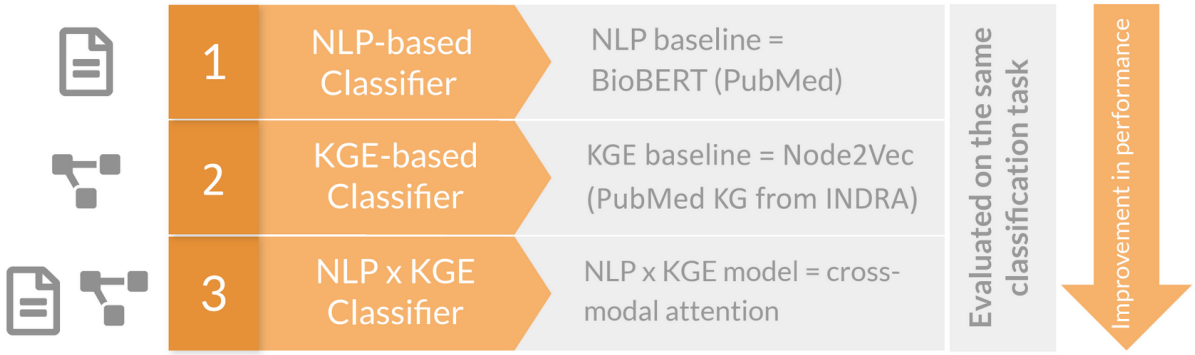


Figure 1: Overview of the hypothesis examined in this Master's Thesis. The three proposed models are trained and evaluated in a shared setting. Moreover, the data sources (PubMed and the Integrated Network and Dynamical Reasoning Assembler, INDRA) as well as the models (BioBERT, node2vec) are outlined in Section 2 and 3. (Image source: own.)

Naturally, the question arises as to whether a biomedical ML model utilizing both text and KG embeddings (KGEs) can lead to increased performance compared to models that only rely on either one of the two modalities. The foundation of this comparison is

a common training dataset containing pairs of text and KG data entries and a shared evaluation framework for all three models in a transfer learning setting, consisting of multiple evaluation tasks. Details on the datasets, models and evaluation strategies are given in Section 3. Overall, the general hypothesis can be stated as follows:

Based on a shared training dataset and evaluation framework, a model utilizing both text embeddings and KGEs (NLP x KGE)[1] *leads to improved performance compared to two baseline models, namely a KGE baseline and an NLP baseline, which only use KG or text data from the training dataset, respectively.*

It is important to note that although Figure 1 is indicating a hypothesized improvement of the KGE baseline model over the NLP-based one, it is not a required outcome of this thesis. The main focus lies on the proposed superiority of the NLP x KGE classifier over both baselines rather than on a comparison of the performances between the two baselines.

1.3 Aims of the Master's Thesis

In order to examine the proposed hypothesis, the following objectives are defined for this thesis, resulting in an overall step-by-step procedure:

1. **Data**: Find and pre-process a suitable large-scale dataset comprising a wide range of biomedical literature, which can be used to train all three model variants. The dataset should contain pairs of text and KG data entries.

2. **KGE baseline method**: Select a KG embedding model (KGEM), which will be used to build the KGE baseline and to obtain the KGEs used in the NLP x KGE model. One subgoal is to find a suitable method for sampling sequential inputs (i.e., a list of entities) from the graph-based KGEM in order to be able to build a cross-modal model later on.

3. **NLP baseline method**: Identify an LM, which will be used to build the NLP baseline as well as to obtain the text embeddings used in the NLP x KGE model. It is important to note that it would be preferable not to train a new LM from scratch on the data identified in **1.**, but rather to find an LM, which was trained on a biomedical literature corpus sufficiently close to the dataset used for the other models in this thesis.

[1] It should be noted that in this thesis, the x in NLP x KGE does not represent the mathematical cross product operator and does not indicate a cross product between the two modalities in any way.

4. **NLP x KGE method**: Choose a suitable approach for implementing a cross-modal NLP x KGE model, which processes both text and KG embeddings. The main goal is to take inspiration from the latest research on multimodal transfer learning architectures in the general domain and apply the same concepts in the biomedical setting of this thesis, possibly leading to an improvement of the state of the art. One additional goal is to design the model without an entity linking component, i.e. the model should require explicit alignments between text and KG entities.

5. **KG baseline construction**: Use the chosen KGEM from **2.** to construct the KG baseline model based on a KG built from the training data collected in **1.**

6. **NLP baseline construction**: Use the identified LM from **3.** to construct the NLP baseline model.

7. **NLP x KGE model construction**: Construct the NLP x KGE model using the dataset from **1.** and the methodology derived from **4.** One optional goal is to make this model publicly available so that it can be used for various transfer learning settings.

8. **Evaluation**: Set up multiple biomedically relevant evaluation tasks and metrics, which can be used to evaluate all three models. Ideally, this will result in an overall benchmark-like setting. Prepare respective task-specific datasets.

9. **Comparison**: Evaluate all three models on the chosen tasks in a benchmark-like setting and compare the model performances.

The subsequent sections of this thesis are dedicated to processing these goals successively. Section 2 is providing the theoretical foundations behind KGEs and NLP, as well as an overview of relevant data sources and state-of-the-art methods needed to tackle the steps outlined in **1-4.** Next, an overview of the final dataset used in this thesis is given in Section 3, followed by a detailed explanation of the three proposed models and descriptions of the implementation and evaluation procedure. The central results of the outlined method are given in Section 4 and discussed in Section 5. Lastly, Section 6 is providing a summary of the central findings and insights of this thesis.

2 Related Work

This section lies the foundation for the central methodological concepts and data sources employed in Section 3. First, Section 2.1 provides an overview of the sources for unstructured text and structured KG data used to construct the dataset for this thesis. Then, outlines of recent state-of-the-art research directions in NLP and NRL are given in Section 2.2 and 2.3, focusing on the approaches central to this work. Lastly, Section 2.4 introduces so-called multimodal Transformers, i.e., the model architecture used as an inspiration for combining text and KG data in the NLP x KGE model (see Section 3.2.3).

2.1 Biomedical Databases

As mentioned in Section 1.1, biomedical literature is forming the basis for most large-scale ML models in the biomedical domain. Typically, the contained information is either used in its plain form (i.e., unstructured text data) or assembled into a set of structured statements, which can be used to build a KG. These two variants are employed to generate the text and KG data used in this thesis. In both cases, PubMed (i.e., the largest biomedical literature database [NCB18]) is chosen as a starting point. While Section 2.1.1 provides a general overview of PubMed, its main components and the resulting text data, Section 2.1.2 introduces the Integrated Network and Dynamical Reasoning Assembler (INDRA) [Gyo+17] used to extract KG data from PubMed.

2.1.1 Unstructured Data in PubMed

PubMed is a freely available literature database for biomedical literature, comprising over 32 million citations, abstracts, and metadata of biomedical literature as of June 2021 [Med21a]. It is maintained by the U.S. National Center for Biotechnology Information (NCBI), which is part of the National Library of Medicine (NLM) at the National Institutes of Health (NIH). Since its launch to the public in 1997, the literature database focuses on collecting citations in biomedicine and various related fields, such as bioengineering, life sciences, behavioral sciences, and chemical sciences. Although several sources are used for the extraction of citations, two components make up the vast majority of PubMed:

1. **MEDLINE**: Comprising over 27 million citations of journal articles as of June 2021, MEDLINE is by far the largest component in PubMed [Med21c]. Between its original launch in 1966 and the public launch in 1997, MEDLINE served as an index of medical articles mainly used by university libraries. Ever since, the number of indexed articles has grown continuously, as shown in Figure 2. MEDLINE

includes citations from over 5,200 journals in 40 languages, which are updated daily. The included journals are selected by the Literature Selection Technical Review Committee (LSTRC), consisting of biomedical experts. A key distinctive feature of MEDLINE is that its citations are indexed with Medical Subject Headings (MeSH) (i.e., key terms from an ontology of standardized biomedical concepts) [NCB18]. A citation typically consists of such terms, together with general metadata such as the title, source and authors of the respective publication. Additionally, many citations include an abstract; however, they typically do not contain the full-text article.

2. **PubMed Central (PMC)**: On the other hand, PMC focuses on a collection of full-text articles, which currently covers over 6 million records [Bio21]. These articles mainly stem from a publisher program deposit with over 2,000 journals. Some of these articles are cross-referenced to their respective MEDLINE entries. Moreover, a third of PMC consists of author manuscripts and digitalization projects of older literature.

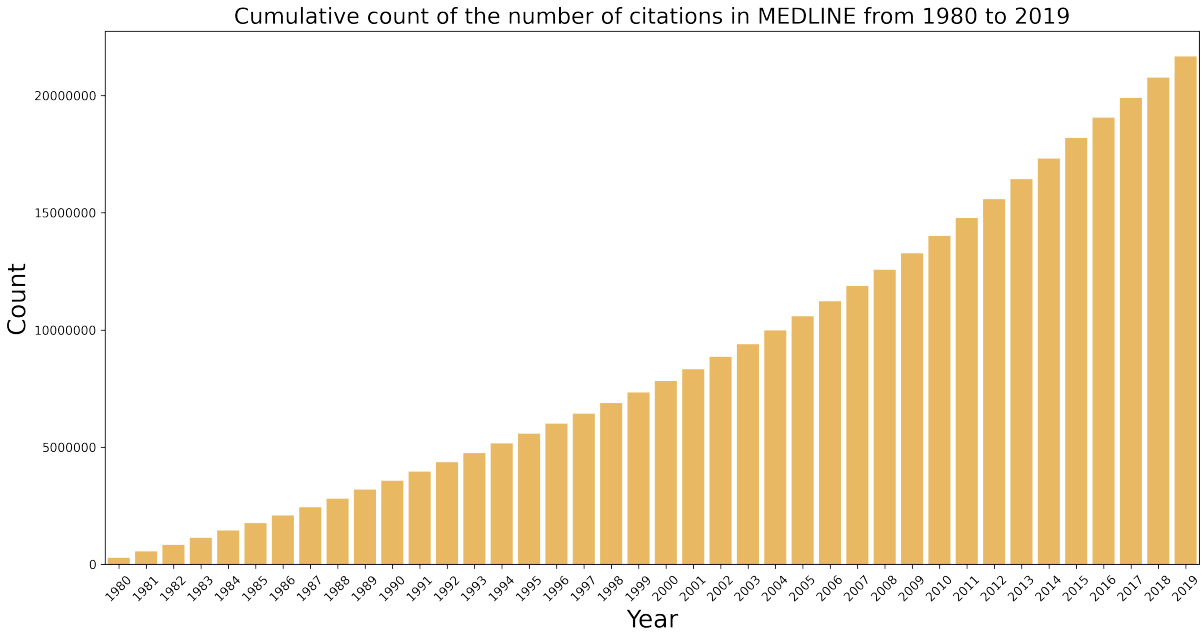


Figure 2: Cumulative count of listed citations in the MEDLINE component included in the PubMed literature database between 1980 and 2019. It is important to note that this overview is restricted to the chosen time span (i.e., the number of citations before 1980 and after 2019 are excluded). As a result, it should be regarded as an exemplary illustration of the growth of MEDLINE rather than a complete depiction of all its citations. (Image source: own, based on the statistics provided in [Med21b].)

2.1.2 Structured Data in INDRA

Essentially, the Integrated Network and Dynamical Reasoning Assembler aims at an automated assembly of biological computational models from descriptions provided in natural

language [Gyo+17], primarily stemming from PubMed. A common use case in such automatically assembled computational models is to simulate biochemical reactions with Ordinary Differential Equations (ODEs) based on the information comprised in biomedical literature. To achieve this goal, INDRA combines numerous extraction sources as well as model assembly approaches and unifies them in a shared three-step software architecture (see Figure 3).

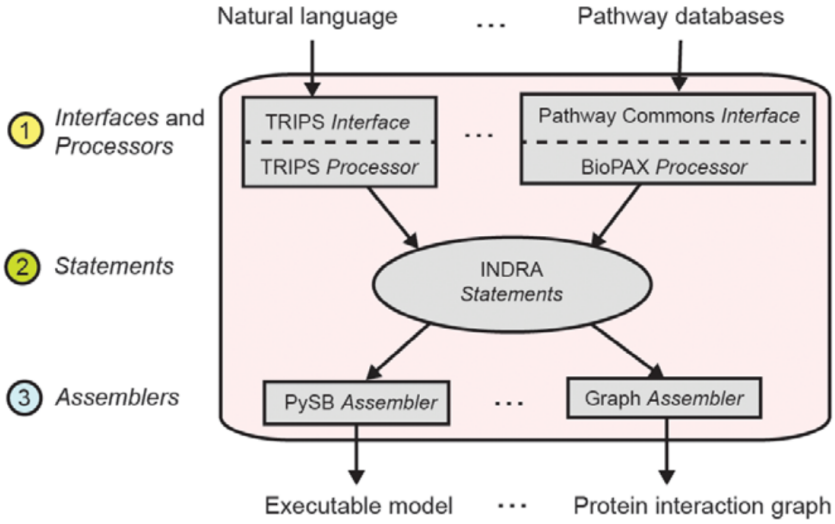


Figure 3: Three layer architecture used in INDRA. First, information from external sources is accessed by source-specific interfaces and processed with its *processors*. The processed information stored in its intermediate format (i.e., in INDRA *statements*), which standardizes the given data from all sources. Lastly, the *statements* are used as a basis for various biological models built by *assemblers*. (Image source: taken from [Gyo+17].)

The first step consists of accessing information from a total of 28 different sources [GB20], which each contain information extracted from biomedical literature. Each source has a different focus (e.g., drugs, chemicals, proteins or more abstract biological processes) and format. An exemplary overview of some of the most important sources (i.e., the ones making the largest contributions) and short descriptions for each listed source are given in Table 1. As highlighted in the table, there are three main groups of extraction sources: reading systems, molecular pathway databases and chemical information databases. The listed reading systems are the REading and Assembling Contextual and Holistic mechanisms from text (REACH) [Val+18] approach, the biological component of the Rochester Interactive Planner System (TRIPS) [All+15] and the Eidos [Sha+19] approach. For molecular pathway databases (i.e., databases containing descriptions of biological processes), the Biological Expression Language (BEL) [Sla14], the Biological Pathway Exchange (BioPAX) [Dem+10] and the SIGnaling Network Open Resource (SIGNOR) [Per+16] are listed in Table 1. Lastly, INDRA includes, among others, the Comparative Toxicogenomics Database (CTD) [Dav+17] and DrugBank [Wis+18] in their chem-

ical information databases. While reading systems are NLP-based and mostly focused on automatic extraction, the molecular pathway and chemical information databases are typically characterized by a higher degree of manual curation.

Type	Name	Description
Reading Systems	**REACH** [Val+18]	Combination of rule-based extraction and CRFs
	TRIPS [All+15]	Context-free grammar processing, inter alia, semantic types and dependency relations
	Eidos [Sha+19]	Based on trigger words, dependency trees and simple word embeddings for entity linking
Molecular Pathway Databases	**BEL** [Sla14]	Subject–predicate–object statements describing biological entities, processes and their relationships
	BioPAX [Dem+10]	Represents, among other things, metabolic and signaling pathways and molecular interactions
	SIGNOR [Per+16]	Manually curated protein-protein interactions
Chemical Information Databases	**CTD** [Dav+17]	Manually curated relations between chemicals, genes and diseases
	DrugBank [Wis+18]	Manually curated drug-drug and drug-target interactions

Table 1: Overview of a few selected statement extraction sources used in INDRA. A full list of all employed sources can be found in [GB20]. Through a respective *processor*, the interactions identified in each source are converted into INDRA *statements*, which possibly also contain text *evidence*. While the molecular pathway and chemical information databases are treated statically, the reading systems are regularly updated and run on the latest content drawn from PubMed (MEDLINE) and PMC [GB21].

By using source-specific *processors*, INDRA intends to standardize and unify the entailed information coming from each source in its second step. As a result, INDRA *statements* form an intermediate representation, which can be used as a basis for many different applications, such as computational models or KGs. Depending on the intended use case, various *assemblers* can be used to represent the desired biological application, as shown in the third step in Figure 3. However, since the dataset used in this thesis is directly constructed from INDRA statements, the assembly process will not be further discussed at this point (see [Gyo+17] for more details on various assemblers).

In its simplest form, statements can be interpreted as a description of biological processes consisting of two entities and a relationship between them. Based on a list of statements, the available information about a given set of entities and their relations can then be used to construct a KG. For most statements coming from reading systems (but also in some instances from molecular pathway and chemical information databases), the extracted information contains an *evidence* attribute, encompassing the natural language description

from which the statement has been extracted (automatically or manually). This text evidence typically consists of one or two sentences from the biomedical article in which a given biological process has been mentioned. Moreover, some statements also contain an *annotation* attribute, which provides additional information on the context in which a biological process takes place. Such context might include, for instance, the disease or species that the biological process refers to. The respective terms are typically taken from controlled vocabularies of various biomedical ontologies.

2.2 Natural Language Processing

The data sources mentioned above offer much potential for the application of sophisticated ML models. Particularly regarding the text data present in biomedical publications in PubMed, Natural Language Processing methods can be used to provide an all-encompassing view on biomedical knowledge contained in large collections of such text sources, so-called text corpora. More specifically, NLP is a sub-domain in ML that deals with the automated interpretation and manipulation of human language. As a result of the ever-increasing size and quality of text corpora (especially in the biomedical domain) and the steady improvement of the computational power of Graphics Processing Units (GPUs) as well as Tensor Processing Units (TPUs), NLP has led to breakthroughs in various Natural Language Understanding (NLU) tasks in recent years. For instance, Microsoft's recent Decoding-enhanced Bidirectional Encoder Representations from Transformers with disentangled attention (DeBERTa) model [He+21] has proven to surpass human baseline scores on the Super General Language Understanding Evaluation (SuperGLUE) benchmark [Wan+19a].

The particular NLP approaches that led to such methodological advances will form a central building block for the methodology used in this thesis. Therefore, the following subsections[2] provide an overview of the respective core concepts needed to understand the approaches presented later on. Starting with an outline of commonly applied preprocessing steps in Section 2.2.1, this subsection intends to gradually lead from NLP foundations (Section 2.2.2 and 2.2.3) to the concrete approaches (Section 2.2.4 and 2.2.5), which will serve as a basis for the models introduced in Section 3.

[2]These subsections are a re-written and shortened version of the NLP foundations presented in the preceding Master's Project [Bal21].

2.2.1 Preprocessing of Text Data

Similar to other ML subdomains, the approaches used in NLP are typically operating on numerical inputs. Hence, raw text data entries present in text corpora need to be converted into a numerical representation beforehand. More specifically, three steps are usually applied to contiguous strings of text to transform them into input features for ML-based NLP models, so-called Language Models: segmentation, normalization and tokenization [JM20, pp. 14–22].

First, segmentation deals with breaking down a long piece of text (e.g., a paragraph) into parts that can be used as chunks of input for an LM. One of the most straightforward segmentation approaches consists of splitting by periods. However, most of the recently published LMs avoid a separate segmentation step by using a fixed input length by which the splits are arranged. Next, normalization aims at creating a more standardized representation of the input data by removing, for instance, special characters. Additionally, the normalization step might include removing unwanted byproducts of web-scraping (i.e., the most common method for generating large text corpora), such as HyperText Markup Language (HTML) syntax. These steps are typically applied in a rule-based manner based on filtering the text with specified patterns, referred to as regular expressions.

Lastly, tokenization is arguably forming the most important preprocessing step. The goal of tokenization is to dissect a given (segmented) piece of text (e.g., a sentence) into its single units, which are then passed as an input sequence to an LM. Each unique unit is then mapped onto a respective numerical representation. Similar to segmenting by periods, the simplest tokenization method involves splitting by whitespace. As a result, words, so-called tokens in this context, become the respective single input units. However, the main problem of this approach lies in the lacking strategy for dealing with out-of-vocabulary (OOV) tokens, which might appear in previously unseen text during inference. Therefore, most modern tokenizers include a maximum number of unique tokens that are mapped onto numerical values (e.g., a vocabulary size of 30,000) and a special token used for other or unknown words. Moreover, modern tokenization methods split by subwords rather than whole words, which proves to be more flexible for compound words since known subwords can be easily identified. The choice of the exact subwords is usually based on picking the ones that appear the most often (or most likely) in natural language. For instance, the WordPiece tokenizer starts with a set of characters (e.g., Latin letters) and iteratively combines two units in the present vocabulary based on a maximum increase of the overall likelihood of a given text corpus. (For a more detailed explanation of the WordPiece tokenizer, see [Wu+16] and [SN12].)

2.2.2 Statistical Language Modelling

Overall, the goal of Language Models is to learn a joint probability distribution of sentences or word sequences, which can be used as a basis for various NLU tasks. This probability distribution aims at distinguishing likely/plausible phrases (e.g., *"Helena loves to swim in the lake when it's sunny outside"*) from unlikely/implausible ones (e.g., *"Helena loves to write her thesis when it's sunny outside"*). More specifically, the probability distribution is (directly or indirectly) learned in a frequentist manner based on examples from text corpora. In its beginnings, NLP approaches have tried to directly model the probability of a given phrase as a product of the conditional probabilities of its single tokens [JM20, p. 31]:

$$P(w_{1:n}) = P(w_1)P(w_2|w_1)P(w_3|w_{1:2})...P(w_n|w_{1:n-1}) = \prod_{k=1}^{n} P(w_k|w_{1:k-1}) \tag{1}$$

This notion assumes that each token in a given sequence is dependent on its predecessors, hence, $P(w_{1:n})$ is describing the joint probability $P(w_1, w_2, ..., w_n)$ of the token sequence $w_1, w_2, ..., w_n$ based on the probability chain rule. Nonetheless, given the creativity and sparsity of natural language, it is extremely challenging to learn conditional probabilities based on a long sequence of predecessors. This is why in one of the earliest successful LMs, namely the n-gram model [JM20, p. 32], the conditional probabilities are restricted to a fixed number of k predecessors per token:

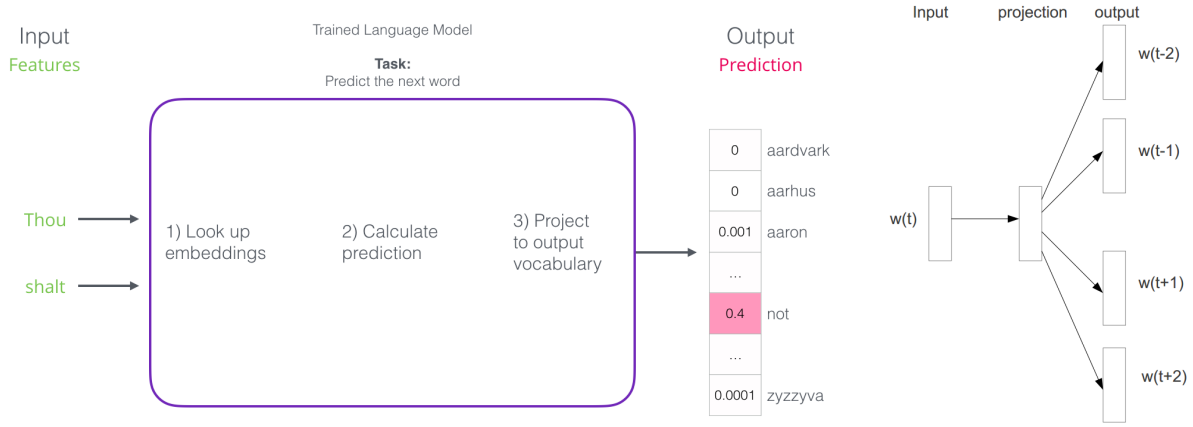
$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-k+1:n-1}) \tag{2}$$

However, such n-gram models have been increasingly replaced with neural LMs over time. One of the first neural LMs was proposed by Bengio et al. in 2003 [Ben+03], and it aimed at representing the probability distribution in an indirect manner through a combination of token feature vectors, so-called token (or word) embeddings. Its core concepts, which are still applied in most (if not all) modern LMs to this day, can be summarized in three steps, as stated in the original publication [Ben+03]:

1. "Associate with each word in the vocabulary a distributed *word feature vector* (a real-valued vector in $\mathbb{R}^m$)" [Ben+03]
2. "Express the joint *probability function* of word sequences in terms of the feature vectors of these words in the sequence" [Ben+03]
3. "Learn simultaneously the *word feature vectors* and the parameters of that *probability function*" [Ben+03]

That being said, most LMs only differ in the approaches used to learn such word embeddings as well as in the method used to combine the embeddings. A general abstract visualization of this method is shown in Figure 4a. One of the reasons for the popularity of this paradigm is its intuitive application of the so-called distributional hypothesis, which states that words that appear in similar contexts are similar in their semantic meaning [JM20, p. 96]. The learned vector space of token embeddings typically intends to represent semantically related words through similar vectors (i.e., vectors with a high cosine similarity).

In particular, the word2vec approach introduced by Mikolov et al. in 2013 [Mik+13a] [Mik+13b] proved to be quite successful at putting this paradigm into practice. In short, two model variants, i.e., the continuous bag-of-words (CBOW) and skip-gram models, introduced respective training objectives that directly target the distributional hypothesis. While the goal of the CBOW model is to predict the middle token in a given sequence of neighboring words, the skip-gram model turns this idea around and aims at predicting the surrounding tokens of a given middle token (shown in Figure 4b). In both cases, the training objectives are used to learn the token embedding projections.



(a) General concept of a neural language model based on word embeddings. (Image source: taken from [Ala18b].)

(b) Skip-gram model. (Image source: taken from [Mik+13a].)

Figure 4: Abstract representation of a neural language model and its concrete realization in the skip-gram model architecture employed in word2vec [Mik+13a] [Mik+13b]. Based on the general idea of projecting tokens onto high-dimensional embedding vectors, word2vec's skip-gram model trains such word embeddings in order to predict the surrounding tokens of a given input token.

The learned token embeddings can be then used as features for various NLU tasks. Such NLU tasks are mainly formulated as classification tasks and can be roughly divided into three categories: document-, sentence- and token-level classification [Kow+19]. For in-

stance, Named Entity Recognition (NER) is a common token-level task that aims at identifying tokens that describe named entities (such as proteins, genes and diseases) in a given text sequence [JM20, p. 153]. Another common sentence-level task is Relation Extraction (RE), which deals with the prediction of specified relationship types (such as *A interacts with B*) for named entities in a given sentence [JM20, pp. 332-334]. Lastly, entity linking is the (token-level) task of correctly mapping a specified text mention to a normalized entity in a database or KG [JM20, p. 416], which includes the disambiguation of similar or identical words in different contexts (e.g., "*Corona* beer" versus "*Corona* infection").

2.2.3 Attention Mechanism and Transformers

Similar to the classic neural LMs outlined in the previous subsection, the goal of the attention mechanism is to learn word embedding representations. In addition to learning embeddings, the attention mechanism overcame two main disadvantages of word2vec and subsequent recurrent neural LM architectures, which arguably led to its widespread success:

1. **Computational inefficiency**: Many recent LMs rely on recurrent architectures (e.g., Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, see [Mik+11] and [Pet+18]). The sequential nature of these LMs makes it impossible to parallelize the training procedure since each component requires the previous one for its calculations. Particularly for longer input sequence lengths, this usually forms the bottleneck in terms of training speed.

2. **Inability to learn long-range dependencies**: On top of the impeded training speed for longer input sequences, most recurrent architectures implicitly assign a higher importance to directly preceding words for calculating the embedding representation of a given token. As a result, this makes it challenging to learn long-range dependencies present between different parts of phrases (especially in long sentences) in natural language. Even though word2vec is not a recurrent model, it effectively faces the same challenge since the training objective is restricted to a fixed and typically small context window (e.g., three to seven words).

This is where the attention mechanism led to a major breakthrough: Instead of relying on a fixed structure for representing the context, the goal is to learn so-called attention weights that determine the relevance of each token in the overall sentence (or word sequence) for the representation of a given token. In more detail, the attention mechanism was originally introduced by Bahdanau et al. in 2015 [Bah+15], and consisted of an RNN-based sequence-to-sequence model for Machine Translation, i.e., a model that creates an

output sequence in a target language based on an input sequence in a source language.

In general, the model architecture consisted of an encoder and a decoder, which each used an RNN to generate hidden state vectors for each token in the input sequence, as well as in the generated output sequence. The introduced attention mechanism referred to how the decoder created the output based on the information from the encoder: Instead of just using the last hidden state h as a basis for generating the i-th output y_i, Bahdanau et al. created a context vector c_i for each output y_i. More specifically, there are three steps involved in generating the context vector (see [JM20, pp. 191–192, 213–214]): comparison, normalization and the calculation of the context vector. First, the comparison step encompasses generating similarity scores between the last decoder hidden state s_{i-1} and each encoder hidden state h_j, typically based on the dot product. Next, all similarity scores are normalized to create the attention weights α_{ij}, using the softmax function. Lastly, the context vector is calculated as a weighted sum of all hidden state vectors of the input sequence of length T_x:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{3}$$

However, a slightly different variant of the the attention mechanism is predominantly used nowadays. The particular variant was introduced by Vaswani et al. [Vas+17] in 2017 and consists of mapping a matrix with the initial embedding vectors X onto query, key and value matrices: $Q = W^Q X, K = W^K X, V = W^V X$ (see [JM20, p. 192] for an additional explanation on the query, key and value analogies). More specifically, the matrices W^Q, W^K, W^V are the parameters of the attention mechanism, which are learned during training. In total, the query, key and value matrices are combined to generate the final embedding representation of each token [Vas+17]:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{4}$$

Even though this notation is different, this attention variant applies the same general idea used in the original attention mechanism. First, the similarity scores are generated for each input pair through the multiplication of Q and K^T. Then, the normalization step is realized through the division by the square root of the embedding dimension $\sqrt{d_k}$ and the application of the softmax function. Lastly, a matrix of context vectors is calculated using the multiplication of the attention weights $a = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})$ with V.

A common technique for visualizing the attention mechanism is to display the learned attention weights by links between different parts of the source and target sentence and use

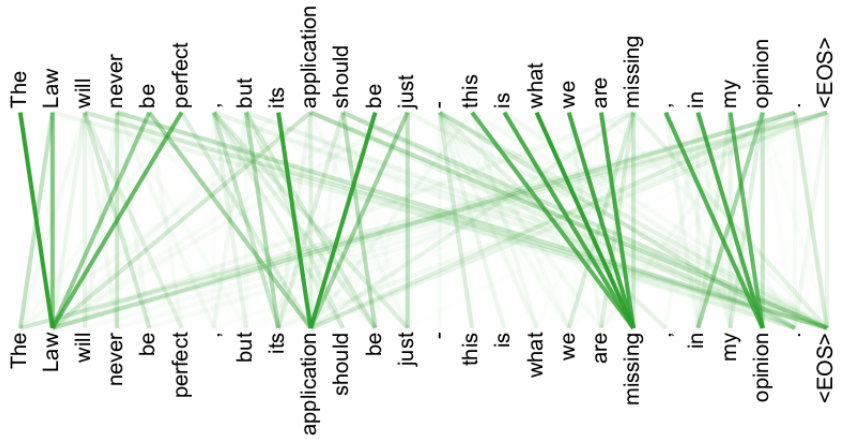


Figure 5: Visualization of the attention weights in the attention mechanism. This example depicts the self-attention employed in the Transformer model architecture introduced by Vaswani et al. [Vas+17]. (Image source: taken from [Vas+17].)

different opacities depending on the numerical value of the respective attention weight. An example of this is shown in Figure 5. By this, it is possible to provide an overview of the learned interdependencies between different parts of a given phrase, which often correlates with how humans view natural language. However, it is important to note that this correlation is not always given and that there are contrasting views on the observed linguistic (in)abilities of attention-based models (see, e.g., [RKR20] for a review).

Together with the previously described variation of the attention mechanism, Vaswani et al. also introduced the Transformer model architecture in the same publication [Vas+17], which forms the basis for various modern LMs. With their introduced sequence-to-sequence architecture, the authors significantly improved the state of the art in Machine Translation. More specifically, the Transformer is a encoder-decoder model (see Figure 6), both parts consist of multiple stacked Transformer layers.

The encoder contains a self-attention and a feed-forward component, whereas the decoder part additionally encloses an encoder-decoder attention component. In self-attention, the attention mechanism is solely applied to the input sequence, i.e., the query, key and value matrices are calculated based on the initial embeddings of the input sequence. On the other hand, encoder-decoder attention aims at learning links between input and output (phrases in the source and target languages from the training dataset in this case) by forming the query and key matrices with the input and the value matrix using the output. Both attention components are realized through multi-head attention, meaning that there are multiple projections of the input X onto multiple query, key, and value matrices: $Q_i = W_i^Q X, K_i = W_i^K X, V_i = W_i^V X$. The resulting representations are then

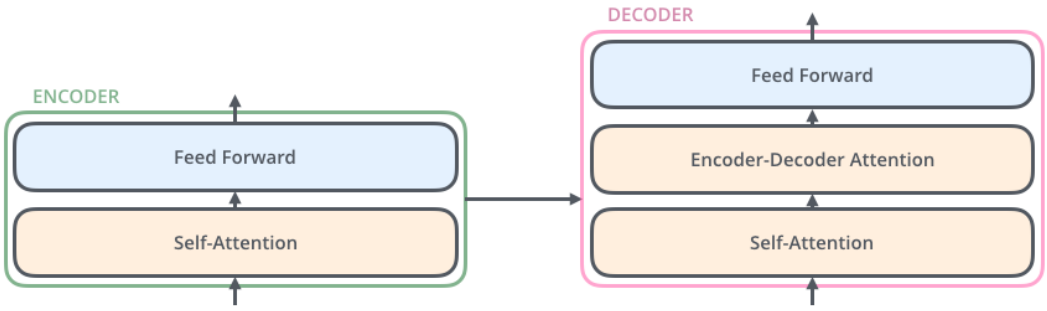


Figure 6: Overview of the Transformer model architecture, which was originally used for Machine Translation. While self-attention focuses on learning links between different parts of the input sequence, encoder-decoder attention aims at learning dependencies between the input and output sequences. (Image source: taken from [Ala18a].)

concatenated and projected to the original embedding dimension using another weight matrix W^O.

2.2.4 BERT: Bidirectional Encoder Representations from Transformers

Although the previously introduced Transformer model architecture was originally intended for Machine Translation, it effectively formed the basis for general LMs, which can be used on various NLP tasks in a transfer learning setting. The Bidirectional Encoder Representations from Transformers (BERT) model is arguably one of the most well-known of such LMs. BERT was introduced by Devlin et al. in 2019 [Dev+19], and it is based on the encoder part of the Transformer model by Vaswani et al. Consequently, BERT is entirely built on self-attention, with the intention of learning links between parts in a given phrase or word sequence rather than between an input and output sequence. Like the original Transformer, BERT is a sequence-to-sequence model with a fixed input length ($n = 512$ tokens). As a result, any input word sequence either needs to be shortened or padded accordingly (using a special `[PAD]` token).

The proposed transfer learning setting consists of two parts, namely the pre-training and the fine-tuning step, as shown in Figure 7. While the pre-training part serves as a basis for learning a general representation of language based on large text corpora, the fine-tuning procedure consists of further training the pre-trained model on labelled datasets. These datasets are typically much smaller than the pre-training corpora and specific to a given NLP task (e.g., NER or RE). In BERT, Devlin et al. use a corpus extracted from Wikipedia as well as the BooksCorpus [Zhu+15] (i.e., a collection of books) for pre-training (3.3 billion words in total). The pre-training examples are generated based on a concatenation of either two consecutive or two randomly sampled sentences from these

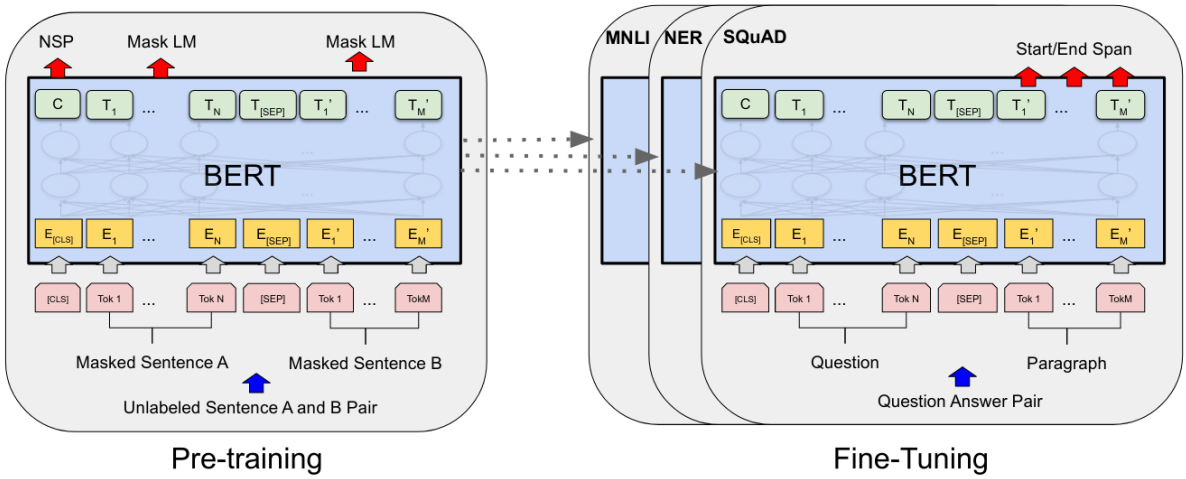


Figure 7: Transfer learning procedure adopted in BERT, consisting of a pre-training and a fine-tuning step. Once the BERT model has been pre-trained on the MLM and NSP objectives, it can be used for multiple NLP tasks (e.g., NER or RE) with minimal architectural changes. (Image source: taken from [Dev+19].)

corpora. Additionally, a special classification (`[CLS]`) token is added in front of the combined sequence, and special separator (`[SEP]`) tokens are added in between and after the two tokenized sentences. For fine-tuning, the authors use a subset of the General Language Understanding Evaluation (GLUE) benchmark [Wan+18]. The training datasets of the chosen tasks range from 2,500 to 392,000 word sequences in size (see [Wan+18] for an in-depth explanation of the datasets and tasks employed in the benchmark). Hence, the pre-training procedure is forming the computationally more intensive part (4 days on 16 Tensor Processing Unit (TPU) chips in this case). In contrast, the fine-tuning procedures usually do not require more than a couple of hours on comparable hardware.

In pre-training and fine-tuning, different training objectives are used to learn a general representation of language and task-specific characteristics, respectively. For pre-training, two objectives are employed in the sequence-to-sequence model: the Masked Language Model (MLM) and the Next Sentence Prediction (NSP) objectives. Masked language modeling is inspired by the Cloze task [Tay53] and aims at correctly predicting tokens, which have been replaced with a special masking (`[MASK]`) token. Devlin et al. randomly mask 15% of the tokens in the input sequence. The partially masked input sequence is first fed through BERT's stacked Transformer layers to generate the predictions for each masked token. Then, each embedding of the output sequence is mapped onto the vocabulary size (30,000 tokens based on a WordPiece tokenizer) using a linear layer with a softmax activation function, which is added on top of the Transformer encoder. This additional layer is also referred to as the language modeling head. On the other hand, the NSP objective intends to distinguish consecutive and randomly combined sentences

in the training dataset. This is achieved by performing binary classification based on the output embedding of the initial `[CLS]` token. As a result, the total loss function of the pre-training procedure consists of the sum of the cross-entropy losses of the MLM and NSP objectives.

The aforementioned `[CLS]` token is also used for generating predictions in fine-tuning tasks, especially in sequence-level classification tasks (e.g., user reviews that are either classified as positive or negative). In such cases, the `[CLS]` token functions as an aggregated representation of the entire output embedding sequence. In analogy to the MLM objective, the predictions are generated by mapping the `[CLS]` vector onto the number of classes of the respective fine-tuning tasks using a linear layer with a softmax activation function. Again, the cross-entropy is forming the loss function that ought to be minimized during fine-tuning.

Due to the widespread success of the pre-training and fine-tuning paradigm as well as the significant increase in performance on the GLUE benchmark, BERT has been adapted to various other domains and languages. For instance, GottBERT [Sch+20a], CamemBERT [Mar+20] and BERTje [Vri+19] are the German, French and Dutch versions of BERT, which have each been pre-trained on language-specific corpora instead of Wikipedia and the BooksCorpus. Common examples of BERT adaptations in the biomedical domain are BioBERT [Lee+20], ClinicalBERT [Als+19] and SciBERT [BLC19]. BioBERT will be of particular interest in Section 3 since it has been pre-trained on the PubMed and PMC corpora discussed in Section 2.1. Lastly, there are also multiple approaches, which aimed to improve the hyperparameter choice and further model design choices of BERT. To give an example, RoBERTa is a BERT-based model, which has led to improved GLUE benchmark scores by leaving out the NSP training objective, improving the masking pattern and using larger batches [Liu+19].

2.2.5 Knowledge Graph Enhanced Transformers

Ever since the advent of the first embedding approaches for large-scale text corpora, combining KGs and word embeddings has been an evident option for integrating factual knowledge into NLP applications. For instance, in 2015, Celikyilmaz et al. used entities from Freebase [Bol+08] related to information about movies to improve word2vec embeddings for semantic tagging [Cel+15], i.e. the task of assigning a specific semantic meaning to various words in a given phrase. In the biomedical domain, one of the most common NLP tasks that benefited from the inclusion of KGs is the extraction of Chemical-Disease Relations (CDRs). Multiple approaches are based on including KGEs learned from the

Comparative Toxicogenomics Database (CTD) to identify possible mentions of relations in text [Zhe+18; Zho+18; Zho+19]. However, such approaches are tailored to one specific NLP task and are hardly generalizable. That is why the following overview is primarily focusing on Transformer-based LMs that are enhanced with KGs since these models can be typically adapted to many different tasks in a transfer learning setting.

Table 2 is showing a comparison of selected KG-enhanced Transformers published in 2019 and 2020. One of the first published approaches on KG-enhanced Transformers was the Enhanced Language Representation with Informative Entities (ERNIE) model [Zha+19] by Zhang et al. This model is not to be confused with the Enhanced Representation through Knowledge Integration (ERNIE) [Sun+19] model by Sun et al., which is based on altering the masking strategy rather than integrating KGs. (For the remainder, ERNIE is referring to the model published by Zhang et al.) Based on the pre-trained BERT model, ERNIE integrates Wikidata [Fou19] KGEs based on information fusion, i.e. a linear combination of KG and word embeddings of tokens that have been mapped to respective KG entities. More specifically, ERNIE extends the MLM and NSP training objectives presented in BERT with a third objective called the denoising entity auto-encoder (dEA), aimed at correctly predicting masked token-entity alignments based on given alignments in the training data.

The trend of using Wikipedia data in combination with aligned Wikidata (or WordNet [Pri21]) entities is continued in several other KG-enhanced Transformers, such as KnowBert [Pet+19], K-Adapter [Wan+20] and Knowledge-enabled BERT (K-BERT) [Liu+20]. It is important to note that the entity linker is not fixed in KnowBert, but learned alongside other model components based on training examples [Pet+19] (therefore still requiring linked entities in the training procedure). In contrast to these three approaches, the Knowledge Embedding and Pre-trained Language Representation (KEPLER) model [Wan+19b] is not directly using plain Wikipedia text data, but rather KG entities from Wikidata. These entities contain metadata in the form of short textual descriptions from the Wikipedia page for a given entity. The overall goal is to minimize both the MLM objective and a KGE loss function based on the sequence embedding representations of the textual descriptions of the entities. Hence, no entity linking procedure is required in KEPLER. Unlike the previously presented approaches, the Contextualized Language and Knowledge Embedding (CoLAKE) model tries to incorporate KG entities through a so-called word knowledge graph (WK graph), which intends to include subgraph structures between the linked entities in a given phrase. Practically, the subgraph structure is learned by concatenating word and KG nodes into a sequence and learning the entity

embeddings with an extended MLM objective, as shown in Figure 8.

Regarding the biomedical domain, the most evident choice for a large-scale text corpus is PubMed. However, finding corresponding entities in a KG is not trivial as in Wikipedia since there are aligned entities by default. Moreover, biomedical approaches tend to use various heterogeneous KGs rather than a unified KG (i.e. Wikidata). One example of a biomedical KG-enhanced Transformer that uses various sources for generating a KG is the biomedical KG-LM (BioKGLM) [Fei+20], which is based on BERT and an in-house KG built from, among others, Reactome [Fab+16] and UniProt [The15]. In BioKGLM, the objective is to correctly predict the (masked) entities based on concatenated token and KG embedding vectors [Fei+20]. Another proposed biomedical KG-enhanced Transformer is BERT-MK [He+20], which also contains a information fusion component similar to ERNIE. However, the major difference is that BERT-MK is integrating subgraphs, rather than isolated entities, into the Transformer. More specifically, this is achieved by sampling a subgraph based on n-hops from a given start entity and representing the subgraph by its adjacency matrix in the model. Furthermore, the so-called triple restoration task, which aims at preserving the sampled subgraph structure in the overall model, is added to the existing pre-training tasks [He+20]. Both BioKGLM and BERT-MK rely on explicit alignments between entities and text.

2.3 Network Representation Learning

Structured data, as for instance present in INDRA, can help to provide a deeper insight into the dependencies and interactions between thousands or up to millions of structured *facts*, resulting in an all-encompassing view on a given field. Such data is typically organized in a *network*, and it can include anything from a group of people and relationships (e.g., [Hin90]), a system of synonyms in linguistics [Pri21], or interactions between chemicals, proteins and diseases in the biomedical domain [LHZ21]. In its simplest form, a network consists of a set of *entities* (also refered to as *nodes*), which are connected pairwise through undirected *edges* [HYL18].

The rich information present in these networks can be used for various applications, such as the classification of different interests of a social media user (*entity classification*), word-sense disambiguation (*entity disambiguation*) or the prediction of candidate drugs (*link prediction*). In their beginnings, these applications relied heavily on hand-engineered features. However, with the advent of Machine Learning, the named tasks were formulated as classification tasks, and the respective features were obtained by applying dimensionality reduction techniques (such as Principal Component Analysis, PCA) on the adjacency

Model	Backbone	Knowledge Integration	Needs An Entity Linker?	General/ Biomedical?	Training Data
ERNIE [Zha+19]	BERT	Information fusion	✓	General	Wikipedia & Wikidata
KnowBert [Pet+19]	BERT	Knowledge attention & contextualization component	(✓)	General	Wikipedia & WordNet
KEPLER [Wan+19b]	RoBERTa	Joint KGE & LM objective	✗	General	Wikipedia & Wikidata
K-Adapter [Wan+20]	RoBERTa	Factual adapter based on relation classification	✓	General	Wikipedia & Wikidata
K-BERT [Liu+20]	BERT	Sentence tree enhanced token sequence	✓	General	Chinese Wikipedia & DBPedia
BioKGLM [Fei+20]	BERT	Entity prediction based on token + entity embeddings	✓	Biomedical	PubMed & Reactome & UniProt
CoLAKE [Sun+20]	RoBERTa	Word knowledge graph structure	✓	General	Wikipedia & Wikidata
BERT-MK [He+20]	ERNIE (BERT)	Information fusion with subgraph conversion	✓	Biomedical	PubMed & UMLS

Table 2: Comparison between central characteristics of the most recent KG-enhanced Transformer LMs. As shown in this table, the presented approaches mainly differ in the methods used for knowledge integration. A majority of the outlined approaches are trained using KG entities that are aligned with Wikipedia text data. All of the presented approaches are built based on either BERT or RoBERTa.

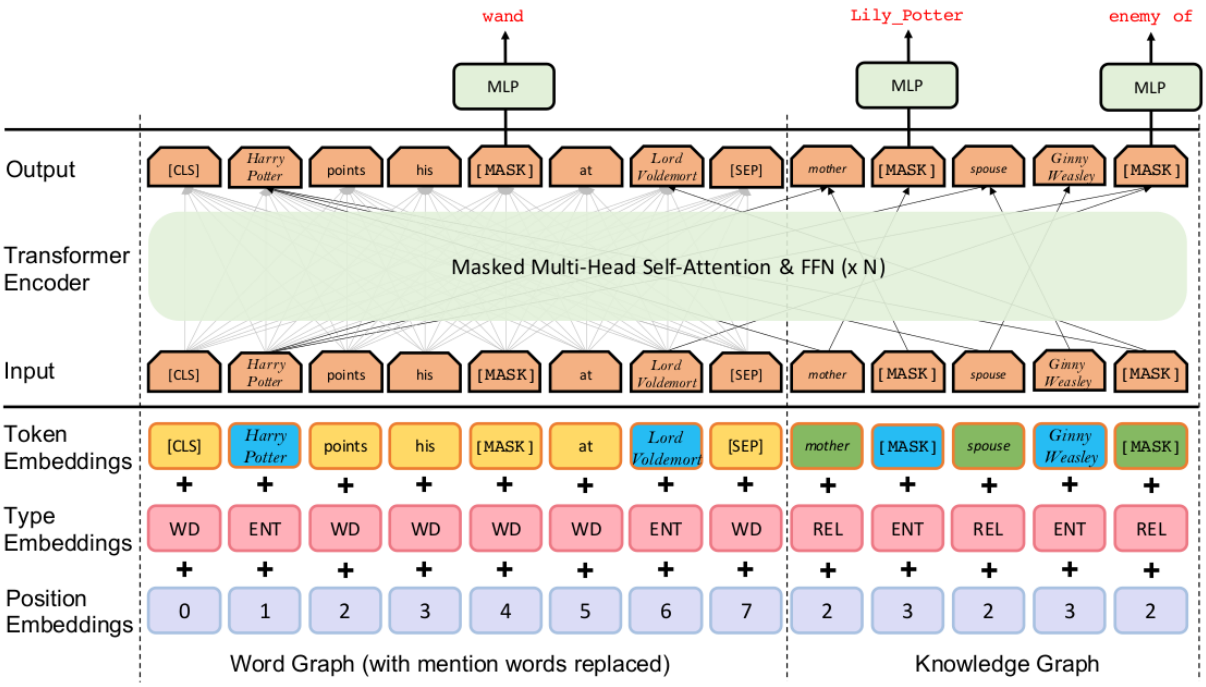


Figure 8: Concatenation of word and entity nodes in CoLAKE [Sun+20]. Based on the pre-trained RoBERTa model, Sun et al. further train the transformer on Wikipedia text with aligned Wikidata entities. The MLM training objective is extended to mask both word and KG entities. (Image source: taken from [Sun+20].)

matrix of a given network [HYL18]. Nonetheless, such methods suffered from a lack of flexibility and scalability. Hence, similar to the evolution of NLP methods, projections of entities onto embedding vectors were proposed to solve this problem. As a result, the field of *Network Representation Learning* (NRL) emerged as its own Machine Learning subdomain.

Due to the variety of domains and backgrounds involved in NRL research, the terminology used in this field is characterized by many synonymous terms as well as concepts with only minor differences (see Table 3 for an overview on key terms and their respective synonyms in NRL). For instance, a network is generally referring to a collection of facts with undirected edges. In contrast, the term *Knowledge Graph* is typically used to describe a set of entities with directed edges of specific *relation* types $r \in \mathcal{R}$. That is why in KGs, facts are usually referred to as *triples* of the form (h, r, t), describing a relation r between a head entity h and a tail entity t. Moreover, in many KG-based applications, the term *Knowledge Representation Learning* (KRL) is often used instead of NRL, and the trained embedding models are referred to as *Knowledge Graph Embedding Models* (KGEMs). Additionally, KGEMs typically learn embeddings not just for the entities but also for the relations r. In practice and the remainder of this thesis, the terms KG and network (just

like NRL and KRL) are used interchangeably.

Term	**Synonyms**	**Description**	**Mathematical Notation**
Entity	Node, Vertex	Individual units in a network/graph that represent anything from a concrete instance to an abstract concept	$e \in \mathcal{E}$ or $h, t \in \mathcal{E}$ or $u, v \in \mathcal{E}$ [3]
Relation	Edge	Links representing a connection between entities	$r \in \mathcal{R}$
Triple	Fact	Knowledge about a relation between a head and a tail entity	$(h, r, t) \in \mathcal{F}$
Network	Undirected (Knowledge) Graph	Collection of nodes and undirected edges between them	$\mathcal{G} = \{\mathcal{E}, E\}$, $\{u, v\} \in E$
Knowledge Graph (KG)	Knowledge Base (KB)[4]	Collection of triples (i.e., nodes with directed edges between them)	$\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{F}\}$
Network Representation Learning (NRL)	Representation Learning	"Learn a mapping that embeds nodes [...] as points in a low-dimensional vector space" [HYL18]	$f_{NRL} : \mathcal{E} \rightarrow \mathbb{R}^d$ [5]
Knowledge Representation Learning (KRL)	Knowledge Graph Embedding (KGE)	"Map [...] entities and relations into low-dimensional vectors" [Ji+21]	$f_{KGE} : \mathcal{E} \cup \mathcal{R} \rightarrow \mathbb{R}^d$ [6]

Table 3: Overview of the the terminology and common synonyms used in NRL, based on [HYL18; Ji+21; LHZ21; Ali+20]. Networks and Knowledge Graphs are describing collections of unspecific/undirected and specific/directed edges for a given set of entities, respectively. Hence, networks (and NRL) can be seen as superordinate generalizations of KGs (as well as the field of KRL).

Similar to the variety in NRL terminology, there are also several strategies for categorizing different groups of NRL modeling approaches. For instance, Hamilton, Ying, and Leskovec list matrix factorization-based embedding approaches such as Laplacian Eigenmaps [BN01] and the Graph Factorization (GF) algorithm [Ahm+13] as well as random walk-based methods like node2vec [GL16] (further discussed in the next subsection) as shallow approaches, since the learned embeddings result in a static embedding lookup table [HYL18]. The authors contrast these models with deep embedding methods such

[3] In some cases (e.g., [HYL18]), $\mathcal{E}$ is used to denote the set of *edges* instead, and the set of entities is referred to as $\mathcal{V}$.

[4] Technically, a *Knowledge Base* is specifying a set of triples based on a specified semantic structure, whereas a *Knowledge Graph* is typically focusing on the graph structure emerging from such a set of triples [Ji+21]. Practically, the two terms are used interchangeably in many publications.

[5] $\mathbb{R}^d$ can be replaced with any other suitable vector space or manifold.

[6] See previous footnote.

as Graph Convolutional Networks (GCNs) [KW17] and the Graph Sample and Aggregate (GraphSAGE) method [HYL17], which dynamically represent a given entity or relation based on its graph neighborhood. On the other hand, Ji et al. propose a more fine-grained categorization, including, among others, linear [Bor+13], Convolutional Neural Network- (CNN-) [Det+18], RNN- [NRM15] and attention-based [Vel+18] models [Ji+21]. Nonetheless, Ji et al. resemble Hamilton, Ying, and Leskovec in their clear progression from standard factorization or ML methods (i.e., static embeddings) to state-of-the-art DL-based embedding approaches (i.e., dynamic embeddings).

Contrary to the NLP domain, more sophisticated DL-based NRL approaches have not established themselves as the predominant choice in most KGE applications yet, and static embedding methods are still widely used. This circumstance is mainly caused by several challenges regarding the scalability of dynamic embedding approaches on large-scale KGs:

1. **Lack of parallelization**: In analogy to classic CNNs, most dynamic embedding approaches operating on graphs are based on aggregating the embeddings of the local neighborhood of a given entity (or relation) to form its embedding vector. As a direct result, it is hardly possible (in fact, in most cases impossible) to partition the graph into smaller subgraphs to learn the embeddings. Hence, the entire KG needs to be loaded into memory for training the KGEM. Large-scale KGs frequently contain hundreds of thousands of entities and up to millions of triples. Therefore, training many modern KGEMs on large-scale KGs requires hardware with hundreds of gigabytes of memory.

2. **Exploding number of parameters**: Many DL-based embedding approaches (e.g., Graph Attention Networks (GATs) [Vel+18]) learn an attention coefficient for each pair of connected entities in a given graph. Hence, the number of trainable parameters is proportional to the number of edges $|E|$, which, in the worst case, is quadratically proportional to the number of entities in the graph ($|E| = \frac{|\mathcal{E}|(|\mathcal{E}|-1)}{2} = \mathcal{O}(|\mathcal{E}|^2)$).

3. **Inefficient inference**: Contrary to static embeddings, which are by definition limited to the entities (and relations) seen in training, dynamic methods can generate new embeddings for previously unseen entities based on their graph neighborhood. However, the entire KG needs to be loaded in memory for this step since the embeddings are calculated at runtime.

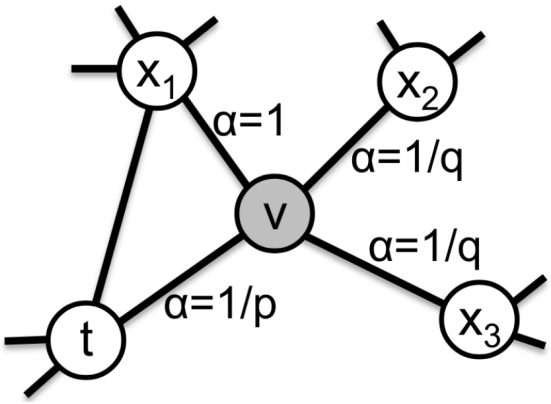


Figure 9: Visualization of the return and in-out parameters (p and q) used to specify the weights α for the transition probabilities π_{vx} in node2vec. Based on a previous transition from t to v, p and q are used to follow an either BFS-like or DFS-like strategy for picking the next edge (i.e., either (v, x_1), (v, x_2) or (v, x_3)). (Image source: taken from [GL16].)

2.3.1 Node2vec

Since node2vec [GL16] serves as a basis for the KG embeddings used in Section 3, a more detailed description of the model is given in the following paragraphs. Node2vec is a static KGE approach that first generates random walks for each node (i.e., entity) in a given graph and then learns embeddings using word2vec's skip-gram model [Mik+13a] on each random walk. The overall method is summarized in Algorithm 1. As a result of this approach, each node of the KG is associated with one embedding vector as well as one random walk. It is important to note that this method is independent of the relation types r of the edges. Hence, it can be applied on (undirected) networks.

As the name suggests, a random walk is a sequence of nodes, which is constructed by randomly traversing a graph starting from a given node u. Rather than just uniformly sampling the next node from a list of neighboring nodes, Grover and Leskovec introduce a probability distribution $P(c_i = x|c_{i-1} = v)$ based on a transition probability π_{vx} and a normalization constant Z for choosing the next node x in a walk, given a current node v:

$$P(c_i = x|c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & \text{if} \quad \{v, x\} \in E \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

The main incentive behind this probability distribution is to allow for more control over the nature of the constructed random walk. More specifically, Grover and Leskovec name two main strategies for sampling neighboring nodes, either Breadth First Search (BFS) or Depth First Search (DFS). While BFS is more tailored towards exploring the local neighborhood of a given start node u (i.e., nodes with a fixed distance to u), DFS aims at a

more global exploration of the (sub)graph by iteratively increasing the distance to the start node for each new node in the random walk. The authors argue that the ideal sampling strategy is a blend of both strategies that is dependent on the concrete application at hand. Therefore, Grover and Leskovec introduce two hyperparameters, p and q, which control the impact of the two complementary approaches on the transition probability π_{vx}. More precisely, for a network with no edge-specific weights, the probability π_{vx} describing the transition from v to any node x is determined by a so-called search bias $\alpha_{pq}(t, x)$, which is dependent on v's predecessor t and the distance d_{tx} between t and x:

$$\pi_{vx} = \alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} & \text{if} \quad d_{tx} = 0, \quad t = x \\ 1 & \text{if} \quad d_{tx} = 1, \quad t \neq x \quad \text{and} \quad \{t,x\} \in E \\ \frac{1}{q} & \text{if} \quad d_{tx} = 2, \quad t \neq x \quad \text{and} \quad \{t,x\} \notin E \end{cases} \tag{6}$$

Since the search bias is formed using the reciprocals of p and q, smaller values of p and q encourage a more locally or globally focused random walk, respectively. A concrete example for the effect of p and q on the overall sampling strategy is shown in Figure 9. Grover and Leskovec list $[0.25, 0.5, 1, 2, 4]$ as the chosen range in which they conduct hyperparameter searches for the optional values of p and q.

Based on the pre-determined values for the transition probabilities π_{vx} (see line 1 in the first procedure in Algorithm 1), the random walks are generated for each node in the graph (see line 6-9, as well as the entire second procedure). The second part of the algorithm (i.e., using word2vec's skip-gram model) leverages the generated random walks to learn the embedding function $f : \mathcal{E} \rightarrow \mathbb{R}^d$ (see line 10 of the first procedure), maximizing the following log-likelihood:

$$\max_f \sum_{u \in \mathcal{E}} logP(N_s(u)|f(u)) \tag{7}$$

More specifically, $P(N_s(u)|f(u))$ expresses the probability of observing the neighborhood $N_s(u)$ (i.e., the list of nodes in the random walk) for a given node u. Similar to word2vec's skip-gram model [Mik+13a], two additional assumptions as well as a simplification are used to generate a computationally feasible maximization problem based on Formula 7:

1. **Conditional independence** (see Formula 8): The probability of observing a neighboring node n_i given the embedding representation $f(u)$ of u is independent of all of its other neighbors. Hence, the conditional probability $P(N_s(u)|f(u))$ can be reduced to the product of the probabilities for the individual neighbors.

Algorithm 1 The *node2vec* algorithm. (Source: taken from [GL16].)

procedure LEARNFEATURES(Graph $G = (V, E, W)$, Dimensions d, Walks per node r, Walk length l, Context size k, Return p, In-out q)
  π = PreprocessModifiedWeights(G, p, q)
  $G' = (V, E, \pi)$
  Initialize *walks* to Empty
  for *iter* $= 1$ **to** r **do**
    for all nodes $u \in V$ **do**
      walk = node2vecWalk(G', u, l)
      Append *walk* to *walks*
    end for
    f = StochasticGradientDescent($k, d,$ *walks*)
  end for
  return f
end procedure

procedure NODE2VECWALK(Graph $G' = (V, E, \pi)$, Start node u, Length l)
  Initialize *walk* to $[u]$
  for *walk_iter* $= 1$ **to** l **do**
    $curr = walk[-1]$
    V_{curr} = GetNeighbors($curr, G'$)
    s = AliasSample(V_{curr}, π)
    Append s to *walk*
  end for
  return *walk*
end procedure

2. **Symmetric relations between nodes** (see Formula 9): The individual likelihoods $P(n_i|f(u))$ are modeled using a softmax expression based on the dot product similarity, normalized by a set of potential source-neighborhood node pairs $\{u, v\}$. Inherently, using the dot product results in a symmetric similarity measure between the embedding representations of two given nodes.

3. **Negative sampling** (see Formula 9): The denominator of Formula 9 intends to normalize the observed similarity between u and n_i. However, given the large size of the set of edges $\mathcal{E}$, using all possible node pairs proves to be unfeasible. Thus, a significantly smaller subset (i.e., $\mathcal{E}'$) is used to approximate the normalization constant. $\mathcal{E}'$ is based on the neighbors n_i as well as a small number (typically between 5-20) of negative samples (i.e., non-neighboring node pairs).

$$P(N_s(u)|f(u)) = \prod_{n_i \in N_s(u)} P(n_i|f(u)) \tag{8}$$

$$P(n_i|f(u)) = \frac{exp(f(n_i) * f(u))}{\sum_{v \in \mathcal{E}'} exp(f(v) * f(u))} \tag{9}$$

The for-loop in line 5 of the first procedure in Algorithm 1 indicates multiple iterations for the given maximization problem. While it is common to run multiple iterations for smaller KGs, it is often computationally impossible to run more than one iteration for large-scale KGs with hundreds of thousands of nodes (specifically for longer random walk lengths l). Given a sufficiently interconnected and large KG, a single iteration can be adequate to learn embedding representations analogous to running one epoch of the word2vec algorithm on a sufficiently large and diverse text corpus. All in all, node2vec can be summarized as a computationally efficient KGEM, which generates an embedding lookup table and a random walk for each node in a given network by applying the word2vec skip-gram algorithm on the generated sequences of nodes.

2.3.2 Text Enhanced Knowledge Graph Embedding Models

Similar to KG enhanced Transformers in NLP (see Section 2.2.5), text data can be used to enhance and improve upon existing KGEMs. Typically, the included text is either present in the form of entity and relation descriptions or textual metadata (e.g., a Wikipedia article) associated with a given entity. Analogous to the inclusion of KG embeddings in NLP, the main challenge for including textual embeddings in KGEMs is to find an approach to combine both modalities in a shared embedding space meaningfully. An overview of the models discussed in the following is provided in Table 4.

One of the first approaches combining both text and KG embeddings was introduced by Wang et al. in 2014 [Wan+14]. The main idea of the proposed approach was to train an alignment model $\mathcal{A}$, which aligns KG embeddings $\mathcal{K}$ coming from a probabilistic extension of the (linear) TransE model [Bor+13] with text embeddings $\mathcal{T}$ from a variant of word2vec's skip-gram model [Mik+13a]. More precisely, Wang et al. defined a log-likelihood-based training objective that aligns the best matching text for a given entity based on either the entity name or a so-called Wikipedia anchor (linking to a page text). Overall, the model was trained by jointly minimizing the losses of the two embedding models combined with the alignment loss: $\mathcal{L} = \mathcal{L}_{\mathcal{K}} + \mathcal{L}_{\mathcal{T}} + \mathcal{L}_{\mathcal{A}}$. The authors validate the performance gain of the combined embeddings on, amongst others, a (binary) triple classification task, which consists of predicting whether a potential candidate triple (h, r, t) is a true triple (i.e., part of the test dataset) or not (i.e., a negative example).

Most other text-enhanced KGs, which have been published after Wang et al.'s approach, share the same idea of using joint training objectives based on both text and KG embedding models. For instance, Xie et al. extend the training objective of TransE (i.e., minimizing $\|\mathbf{h} - \mathbf{r} + \mathbf{t}\|$ for a given triple (h, r, t) [Bor+13]) to combinations of text and KG embedding vectors for $\mathbf{h}$ and $\mathbf{t}$ in their Description-Embodied Knowledge Representation Learning (DKRL) model [Xie+16]. Another example is demonstrated in the Semantic Space Projection (SSP) approach by Xiao et al., building upon a shared loss based on the sum of a TransE-like loss function for learning structural KG properties and a Latent Semantic Analysis (LSA) based topic modelling objective for the available text data [Xia+17]. Both of the named appraoches were validated on Knowledge Graph Completion (KGC) tasks, which intend to find the correct matching entity for an incomplete triple $(?, r, t)$ or $(h, r, ?)$. On the contrary, in many of the more recent applications, text-enhanced KGs are constructed and applied in text-to-data generation settings. For such settings, encoder-decoder architectures (such as extensions of Vaswani et al.'s Transformer [Vas+17]) are commonly used to encode the KG into an embedding space first, in order to then decode the resulting representation into text (e.g., the Knowledge-Grounded Pre-Training (KGPT) [Che+20a] and Entity-to-Description (ENT-DESC) [Che+20c] approaches).

2.4 Multimodal Transformers

Section 2.2.5 and 2.3.2 have outlined several approaches, which benefit from a joint training procedure on two modalities, namely on unstructured text and structured KG data. Naturally, this concept can be extended to any pair or set of modalities using the same methodology. Hence, such applications serve as further inspiration for the methodology developed in this thesis. More specifically, the Transformer architecture is of special interest in this context, not only due to its widespread success in multiple ML application domains (e.g., NLP [Dev+19], computer vision [Dos+21] and bioinformatics [Ji+20]), but also its straightforward architecture that easily allows for combinations of multiple modalities. Intuitively, the weighted average present in the attention mechanism of the Transformer can act as a learnable weighting component, which evaluates how single parts of different modalities contribute to the overall contextual representations. That is why in the following, an overview of recent multimodal Transformers is given (see Table 5).

In the general domain, text, vision (i.e., images and/or videos) and audio are the three dominating modalities for multimodal Transformers. Overall, multimodal Transformers can be grouped into either *single stream* or *two-stream* architectures, depending on

Model	Backbone	Text Integration	Requires Linking?	General/ Biomedical?	Validation Data
KG and Text Jointly Embedding [Wan+14]	TransE	Combined loss function for text, KG and alignments	✓	general	Freebase/ Wikipedia
DKRL [Xie+16]	TransE	Extend TransE objective to text and KG embeddings	✓	general	Freebase 15K (FB15K) [Bor+13]
SSP [Xia+17]	TransE	Combine TransE and LSA objectives	✗	general	FB15K
KGPT [Che+20a]	GAT	Train on text-subgraph pairs for data-to-text generation	✗	general	Web Natural Language Generation WebNLG [SG18]
ENT-DESC [Che+20c]	GCN	Encoder-decoder model for data-to-text	✓	general	Wikidata/ Wikipedia

Table 4: Comparison between central characteristics of selected text-enhanced KGEMs, analogous to the overview of KG-enhanced LMs given in Table 2. The main idea behind most text-enhanced KGEMs is a joint training objective that combines losses from textual and KG embeddings, as well as the loss of an optional alignment model. Notably, all of the identified approaches are applied using data from the general domain, rather than the biomedical domain.

whether they are using a concatenation of modalities or a transformation between them [Kam+21]. For instance, one of the first two-stream architectures, MulT, introduced by Tsai et al. in 2019 [Tsa+19], tried to combine the three named modalities through three so-called crossmodal Transformers. Each modality-specific crossmodal Transformer consisted of two crossmodal attention components, which each modeled transformations between a given target modality and one of the other two source modalities. More specifically, the transformations were modelled as latent adaptations based on the attention mechanism in which the query matrix Q_α was formed using the target modality α, and the key and value matrices (K_β and V_β) were constructed using the source modality β [Tsa+19]:

$$\beta \rightarrow \alpha = softmax(\frac{Q_\alpha K_\beta^T}{\sqrt{d_k}})V_\beta \tag{10}$$

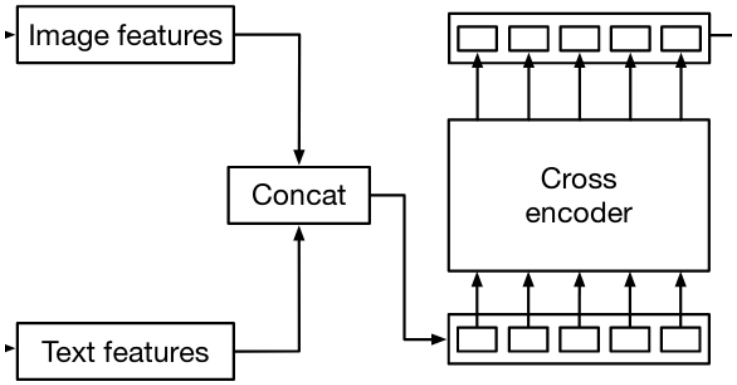


Figure 10: Excerpt from the single stream MDETR model architecture, based on a concatenation of text and image embeddings in a *cross encoder*. The text and image features are generated using a pre-trained textual Transformer and a CNN with a subsequent flattening procedure, respectively. (Image source: taken from [Kam+21].)

For example, the vision-specific Transformer (i.e., for sequences of images V) consists of two latent adaptations from language (i.e., text) to vision $L \rightarrow V = softmax(\frac{Q_V K_L^T}{\sqrt{d_k}})V_L$ and audio to vision $A \rightarrow V = softmax(\frac{Q_V K_A^T}{\sqrt{d_k}})V_A$. The intuition behind these transformations is that two contextualized vision representations are formed using (learned) weighted averages of text and audio data for each image in the vision sequence, respectively. This approach was validated on, amongst other datasets, the Carnegie Mellon University Multimodal Opinion Sentiment and Emotion Intensity (CMU-MOSEI) dataset [Bag+18], consisting of a sentiment analysis task on combined text, audio and video data. Further approaches such as the co-attention module used in the Vision-and-Language BERT (ViLBERT) model [Lu+19] or the cross-modality encoder introduced in the Learning Cross-Modality Encoder Representations from Transformers (LXMERT) approach [TB19] apply the same idea to binary combinations of vision and text data.

However, cross-modal representations in Transformers can also be approached from another direction, namely using single stream methods, which flexibly combine different modalities into a joint input sequence in one Transformer. Such single stream methods are typically based on concatenations of sequences of inputs from different modalities. As a result, such a heterogeneous input sequence can be used to form contextualized representations for a given item in a sequence not only based on other items from the same modality but also from other data sources, using the same attention mechanism (i.e., a shared weighted average across multiple modalities). For instance, the Universal Image-Text Representation (UNITER) architecture by Chen et al. [Che+20b] is based on feeding a concatenation $(\mathbf{w}, \mathbf{v})$ of text tokens $\mathbf{w} = \{\mathbf{w_1}, ..., \mathbf{w_T}\}$ and Region of Interest (ROI) features $\mathbf{v} = \{\mathbf{v_1}, ...\mathbf{v_K}\}$ extracted from image data as the combined input to a

Model	Backbone	Multimodal Integration	Single Stream or Two-Stream?	General/ Biomedical?	Validation Data
MulT [Tsa+19]	Transformer	Latent adaptations between target and source modalities	two-stream	general	CMU-MOSEI
ViLBERT [Lu+19]	BERT	Latent adaptations in co-attention modules	two-stream	general	Common Objects in Context (COCO) [Che+15]
LXMERT [TB19]	Transformer	Latent adaptations in cross-modality encoders	two-stream	general	Visual Question Answering (VQA) v2.0 [Goy+17]
UNITER [Che+20b]	BERT	Concatenation of ROI image and text features	single stream	general	VQA, COCO
MDETR [Kam+21]	Transformer	Concatenation of CCN-based image and text features	single stream	general	COCO
DALL·E [Ram+21]	GPT	dVAE encoder and GPT decoder	single stream	general	COCO

Table 5: Comparison between central characteristics of multimodal Transformers, analogous to the overviews of KG-enhanced LMs and text-enhanced KGEMs provided in Table 2 and 4. Similar to the list of text-enhanced KGEMs, and contrary to the KG-enhanced LMs, this collection of multimodal Transformers is limited to approaches in the general domain.

Transformer. The original MLM training objective is extended with the Masked Region Modeling (MRM) objective for predicting masked ROI features. Moreover, the NSP objective is replaced by the Image-Text-Matching (ITM) task, which aims at distinguishing coherent text-image pairs from randomly paired ones. Another more recent example is the Modulated Detection Transformer (MDETR) model [Kam+21] shown in Figure 10. More specifically, MDETR concatenates flattened image features (that were previously processed by a CNN) with text features, which are already passed through another textual Transformer beforehand. The combined sequence is then used as the input to the so-called cross encoder (depicted on the right in Figure 10).

Lastly, it is important to note that most of the previously mentioned models focus on

the encoder part of the Transformer architecture, similar to BERT. However, a generative model including both an encoder and a decoder can be the more convenient choice in some contexts. For instance, the recent DALL·E model [Ram+21] (in this case, the model name is not representing an acronym, but rather "a portmanteau of the artist Salvador Dalí and Pixar's WALL·E" [Ope21]) contains both an encoder and a decoder component to process concatenated image and text data. More precisely, the encoder consists of a discrete variational autoencoder (dVAE) [Rol17] for image data and a simple tokenizer for text, whereas the decoder is based on the Generative Pre-trained Transformer (GPT) model (i.e., a model architecture using the original Transformer decoder rather than the encoder) applied to the encoded and concatenated input. With this architecture, Ramesh et al. were able to generate images based on a textual description (see [Ope21] for example queries such as "an armchair in the shape of an avocado").

3 Methodology

This section describes the methodology developed in this thesis used to test whether a biomedical ML model utilizing both text and KGEs can lead to increased performance compared to models that only rely on either one of the two modalities. As stated in Section 1.2, testing the hypothesis consists of comparing three models in a shared experimental setting (consisting of the same dataset and evaluation method, as shown in Figure 11a), one using both text and KG data and two baselines using only either one of the modalities (see Figure 11b).

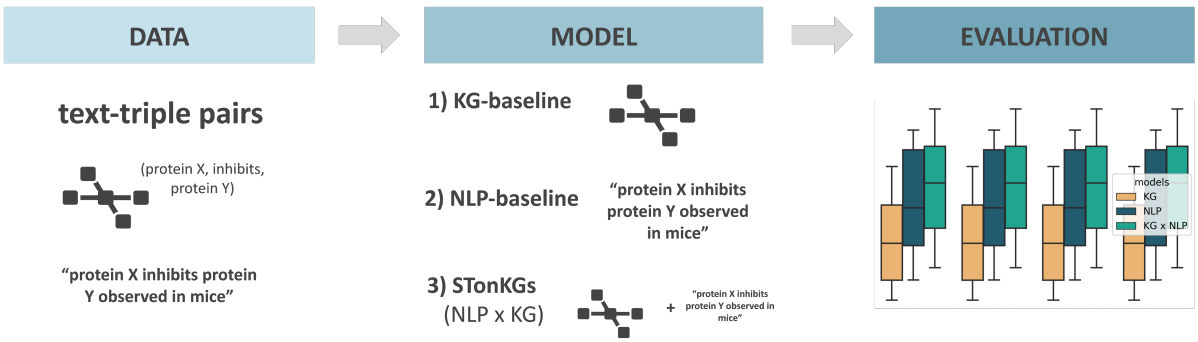
To test the hypothesis, a step-by-step procedure was defined in Section 1.3. Hence, the following subsections intend to subsequently address the items identified in the list of aims of this thesis. First, an overview of the employed dataset and some of its main characteristics is given in Section 3.1. Next, Section 3.2 is providing an in-depth explanation of the three model architectures used in this approach. More specifically, Section 3.2.1 and 3.2.2 are dedicated to the baseline models, i.e., the NLP baseline built on top of BioBERT and the KGE baseline (referred to as the KG baseline in the following) based on node2vec. Section 3.2.3 then introduces the novel NLP x KGE method designed for combining both text and KG data in this thesis: the Sophisticated Transformer Trained on Biomedical Text and Knowledge Graphs (STonKGs). After the elaboration of the model architectures, Section 3.3 describes the method used for evaluating and comparing the performances of the three models in a transfer learning setting. Lastly, Section 3.4 is providing concluding remarks on the implementation of the proposed approach of this thesis.

Moreover, it is important to note that this thesis goes hand in hand with a planned publication that is written in parallel to the thesis[7]. Since both the thesis and the publication refer to the same approach, certain sections from the publication might serve as a basis for sections in the thesis and vice versa. In addition, figures, as well as tables, are shared in the two manuscripts.

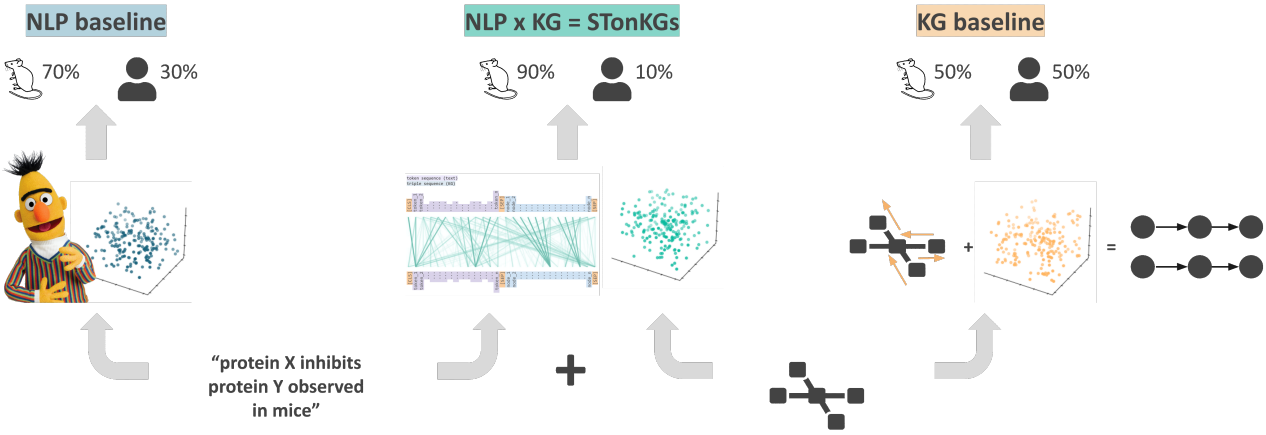
3.1 INDRA Data

Due to the nature of the proposed experimental setting, the dataset that ought to be used by all three model architectures has to ideally contain pairs of text and KG data entries

[7] As of August 18th, a preprint has been published [Bal+21] (https://www.biorxiv.org/content/10.1101/2021.08.17.456616v1, accessed on August 18th, 2021), and the publication is undergoing a review process for journal submission.



(a) The three models are trained and evaluated in a shared experimental setting, consisting of a common dataset of text-triple pairs and an evaluation benchmark.



(b) For each text-triple pair, the two baseline models use only one of the two modalities, whereas STonKGs is utilizing both.

Figure 11: Overall workflow of the methodology used to test whether a model utilizing both text and KG data (i.e., the proposed STonKGs model) can outperform two baselines that are trained on either one of the modalities. (Image source: own.)

(i.e., triples from a KG), referred to as *text-triple pairs* in the following. Using paired entries, contrary to a text dataset with a loosely coupled KG dataset, helps to increase the comparability of model performances when evaluating on different modalities and isolating differences in performances to the change in modalities rather than the nature of the modality-specific dataset subsets.

Hence, the INDRA statements (see Section 2.1.2) consisting of triples and associated text evidence (and in some cases, further annotations) are well suited for the chosen experimental setting. It is important to note that as of July 2021, the complete dataset of all INDRA statements curated by the INDRA labs team at Harvard Medical School (HMS) used in this thesis is not publicly available. More specifically, the dataset used in this thesis is a dump of INDRA statements generated by the INDRA labs team on April 28th 2021, based on applying INDRA to all full-text articles from PubMed. Figure 12 is show-

```
{"type": "Phosphorylation", "sub": {"name": "Prkd1", "db_refs": {"TEXT": "PKD1", "EGID": "18760", "UP": "Q62101"}}, "belief": 1, "evidence": [{"source_api": "rlimsp", "pmid": "28549111", "text": "Interestingly, TAC affected cardiac function only mildly in diabetic mice, which was accompanied by normalization of phosphorylated PKD1, glucose uptake, and cardiac energy status.", "annotations": {"agents": {"coords": [null, [132, 135]]}, "trigger": {"coords": [117, 130]}}, "context": {"species": {"name": "10090", "db_refs": {"TAXONOMY": "10090"}}, "type": "bio"}, "text_refs": {"PMID": "28549111"}, "source_hash": 3023286503622115434}], "id": "197b6632-7cb8-425f-94f2-27fb3e3f6486"}

{"type": "Activation", "subj": {"name": "AMP", "db_refs": {"PUBCHEM": "15938965", "TEXT": "AMP"}}, "obj": {"name": "energy metabolism", "db_refs": {"GO": "GO:0006091", "TEXT": "energy metabolism"}}, "obj_activity": "activity", "belief": 1, "evidence": [{"source_api": "reach", "text": "AMP activated protein kinase (AMPK) is a key regulator of energy metabolism; its activity is regulated by a plethora of physiological conditions, exercises and many anti-diabetic drugs.", "annotations": {"found_by": "Positive_activation_syntax_1_verb", "agents": {"coords": [[0, 3], [58, 75]]}}, "epistemics": {"direct": false, "section_type": null}, "text_refs": {"PMID": "18677519"}, "source_hash": 6675431266515820840}], "id": "8efb205a-e982-4348-8f66-72c9682786a4"}

{"type": "Activation", "subj": {"name": "NSC305787", "db_refs": {"TEXT": "NSC305787"}}, "obj": {"name": "EZR", "db_refs": {"UP": "P15311", "HGNC": "12691", "TEXT": "ezrin"}}, "obj_activity": "activity", "belief": 1, "evidence": [{"source_api": "reach", "text": "Therefore, we are the first to demonstrate that NSC305787 and NSC668394 directly target the anti-metastatic properties of ezrin.", "annotations": {"found_by": "Positive_activation_syntax_1_verb", "agents": {"coords": [[48, 57], [122, 127]]}}, "epistemics": {"direct": false, "section_type": null}, "text_refs": {"PMID": "18601717"}, "source_hash": -2460662848419211313}], "id": "7c1e669d-731b-4aaf-aec9-bc52d7fac700"}
```


Figure 12: Example of INDRA statements contained in the dataset used in this thesis. The generated INDRA data dump is provided as a JavaScript Object Notation (JSON) line (in short, JSONL) file, in which each line consists of a JSON object containing all the information for a given statement. This example demonstrates three different statements containing triple information (highlighted in blue), text evidence (highlighted in purple) as well as an annotation for the first statement (highlighted in orange). (Image source: own.)

ing an excerpt from the dataset, containing three statements (i.e., text-triple pairs) with triple, text and annotation information.

The following subsections aim at providing a closer understanding of the nature of this dataset, as well as the pre-processing steps applied to it. More specifically, the first two subsections intend to explain some of the characteristics of the triples and the text evidences associated with them. The last subsection further specifies the included annotations in the INDRA dataset, which are central for the evaluation procedure (see Section 3.3).

3.1.1 Triples

The original data dump used as a starting point for this thesis contained **35,150,093 statements**. However, INDRA is not only based on reading systems but also other biological databases (see Table 1). As a result, some statements do not necessarily contain any text (or incorrectly generated text, e.g., "No evidence.") for a given triple. Thus, these statements were excluded from further use. Moreover, many statements come from reading systems with triples containing so-called ungrounded nodes (i.e., nodes that are not normalized against any known ontology). Constructing a KG with such nodes is problematic since the lack of normalization is causing a potentially redundant and ambiguous graph. For instance, without a normalization procedure, the nodes `A`β and `amyloid-`β would be modelled as two separate nodes with distinct connections in the graph, even though both terms are referring to the same peptide. Hence, all statements that con-

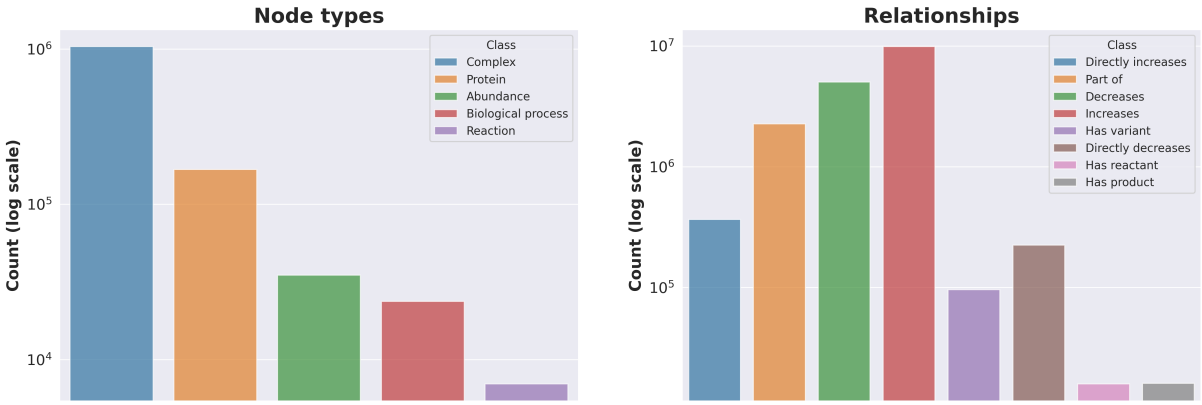


Figure 13: Classes of node and relationship types in the BEL graph constructed from the filtered INDRA statements. Notably, the largest share of nodes are complexes, which are multiple tightly interconnected biological entities merged into a single node. (Image source: own.)

tain ungrounded nodes are filtered out as well. The resulting statements are grounded in normalized terminology stemming from a total of 11 different ontologies, with the majority coming from the Human Genome Organisation Gene Nomenclature Committee (HGNC) [Twe+21] and PubChem [Fu+15] (including Chemical Entities of Biological Interest, ChEBI [Has+13]) ontologies.

Next, a KG using the BEL format (more specifically, using the PyBEL converter [HKE18]) was constructed from the set of statements resulting from the first two filtering steps. This BEL graph (referred to as the INDRA KG in the following) is directly built on the grounded nodes and their relationships described in the triples from the filtered INDRA statements. As a result of the BEL format, there are several relation and node types (not to be confused with the set of unique nodes), as shown in Figure 13. While node types such as proteins or biological processes describe isolated entities, complex nodes (the most common node type in the INDRA KG) refer to a set of multiple linked biological concepts. The use of such complex nodes plays a crucial role in maintaining a computationally manageable size of nodes in the INDRA KG (see Section 2.3 for remarks on the challenges that arise when scaling KGEMs to large-scale KGs). However, one remaining major problem of the constructed KG is the presence of isolated nodes or other small graph components that are not part of the main component. Therefore, all triples that are not part of the largest component in the INDRA KG are filtered out, resulting in a total of **13,609,994 triples** (text-triple pairs), eight different relation types and 174,534 unique nodes.

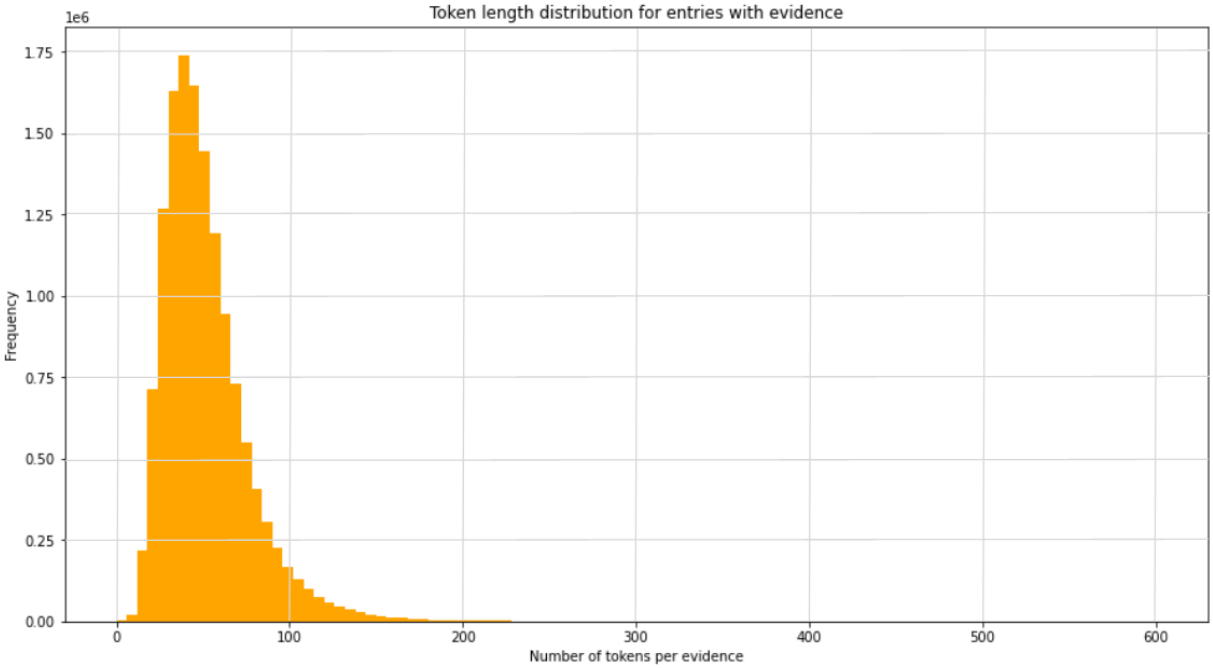


Figure 14: Histogram for the token length distribution of the text evidences contained in the filtered INDRA statements. The number of tokens in each text evidence contained in an INDRA statement is determined by tokenizing the evidence using the BioBERT tokenizer. The resulting tokens in each evidence are counted, counts greater than 600 are filtered out, and the remaining counts are grouped into a total of 100 bins of equal width. The y-axis describes the absolute frequencies of the number of evidences for each bin (in millions). On the other hand, the x-axis specifies the ranges for the number of tokens covered by each bin. (Image source: own.)

3.1.2 Text Evidences

In the pre-processed 13,609,994 text-triple pairs, the contained text evidences were typically extracted at sentence level from the respective PubMed article that a given INDRA statement is based on. As a result, each evidence usually consists of one sentence. Given the specific writing style of scientific publications, this sentence is typically longer than the average English sentence in other text sources such as social media or books. A manual inspection of a small subset of the INDRA dataset revealed that a few text evidences contain more than one sentence (up to five sentences). Moreover, it is important to note that some of the readers in INDRA already applied some basic data cleaning steps to the text data (e.g., removing references). However, those steps are not necessarily the same across all readers. Hence, some textual evidences might have been pre-processed differently than others. To avoid further inconsistencies, no additional cleaning steps (such as case folding or lemmatization) were applied to the text data.

Since the dataset of this thesis is serving as input to Transformer-based models with a fixed input length, it is of crucial importance to assess the number of tokens in each

evidence beforehand to avoid excessive truncation of the input sequences. Figure 14 shows the token length distribution of the text evidences in all pre-processed text-triple pairs. More specifically, the token length of each evidence was determined based on tokenizing the text and counting the resulting number of tokens using the tokenizer of the BioBERT model since the same tokenizer was also used for the NLP-baseline and the proposed multimodal Transformer later on (see Section 3.2.1 and 3.2.3). As depicted in Figure 14, the vast majority of evidences is shorter than 200 tokens. Notably, the resulting distribution is an apparent unimodal distribution rather than a multimodal one, having its peak at around 50 tokens (equating to roughly 30-40 words). Moreover, only a few examples have a meager number of tokens (e.g., less than 20 tokens). Hence, no length-specific filtering was applied to the data, such as removing text-triple pairs in which the number of tokens in the text is lower than a specific threshold.

3.1.3 Annotation Types

As shown in the first example in Figure 12, some INDRA statements contain an additional annotation field, which provides further information about the context in which a given statement has been extracted from. These contexts are either manually annotated or provided by the INDRA processors (i.e., either included as metadata in a biological database or extracted from text with a reading system). It should be pointed out that only 1,507,150 (11.07%) out of the pre-processed 13,609,994 text-triple pairs contain an annotation. Furthermore, even though there are 31 different types of annotations, most of them are not biologically relevant, or there are too few examples per unique annotation in the respective annotation type. However, four annotation types are relevant for the later evaluation procedure (see Section 3.3):

1. **Cell line** (i.e., "a cell culture selected uniformly from a cell population from a usually homogeneous tissue" [Mer21]) ($n = 19,108$ many examples): Provides information on the specific cell line in which a given triple was observed in, which is specifically important for the interpretation of biological experiments.

2. **Disease** ($n = 17,046$): Names the disease in which a certain triple was detected in, mostly covering cancer-related conditions. The disease context can help understand why a given biological interaction characterized by a triple happens in some cases and not others.

3. **Location** ($n = 14,092$): Specifies the cellular location in which the biological process described by a triple occurs in. This information can help to retrace the effects of a given process in larger organisms.

4. **Species** ($n = 1,339,057$): Names the species in which a certain biological observation encompassed by a triple was made. This context is essential in drug discovery processes (see Section 1.1), in which differences between animal test subjects and human patients need to be estimated precisely.

It is important to note that all the annotations are open-ended in theory (i.e., there is no fixed set of classes from which an annotation has to be chosen). Practically, however, most annotations are grounded to an ontology (like the nodes in the INDRA KG). Hence they are limited to the terms listed, which, as a result, is still a considerable number of classes. Therefore, the later evaluation procedure only operates on small subsets of these annotation-specific datasets since the number of classes needs to be lowered to a number that is appropriate for model evaluation beforehand (see Section 3.3 for more details).

3.2 Models

This subsection intends to explain the three proposed model architectures used to test the hypothesis in-depth. The two key motivations behind the design of the experimental setting employed in this thesis are **1.** to enable a high degree of comparability between the data sources and evaluation procedures and **2.** to establish a generalizable evaluation procedure that can help to assess the transferability of the observed performances to future biological use-cases. Transfer learning is a suitable ML paradigm that addresses both requirements:

1. **Unified experimental setting:** The two-fold pre-training and fine-tuning procedure used in transfer learning settings allows for adding an abstraction layer to the overall model comparison. More specifically, the initial pre-training step can be applied to large amounts of unlabelled data, offering the models to learn task-independent embeddings of text and (or) KG data first. The subsequent fine-tuning step allows for transferring the learned general representations of the three models on much smaller labelled datasets by adapting the models' weights using task-specific loss functions.

2. **Assessment of the generalizability of model performances:** The clear distinction between pre-training and fine-tuning allows for the incorporation of multiple fine-tuning tasks in a benchmark-like evaluation setting. The pre-trained models can be repeatedly adapted to different biological use-cases, which can help to provide more insights on the strengths and weaknesses of the models and more evidence for either supporting or rejecting the hypothesis.

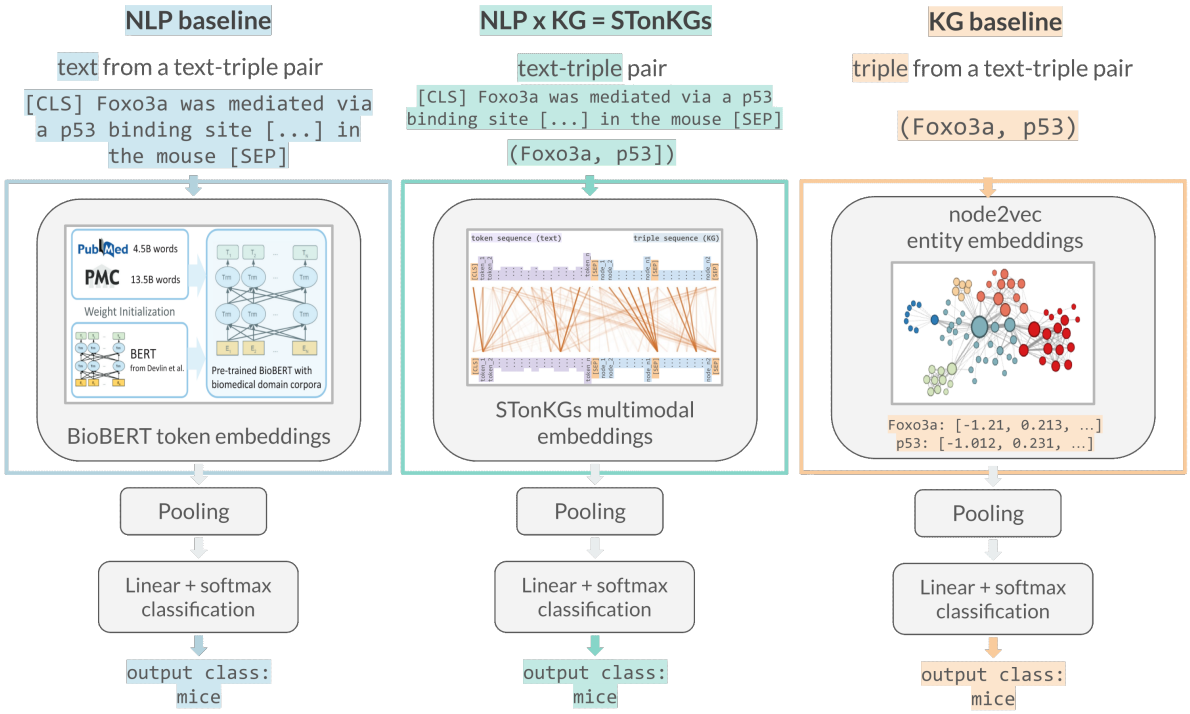


Figure 15: Overview of the three proposed model architectures (contained in the blue, green and orange boxes) as well as their input sources (above the boxes) and the consistent architectural change applied in the fine-tuning setting (below the boxes). (Image source: modified based on [Lee+20], [Vas+17] and [GL16].)

The three proposed model architectures explained in the following have been designed explicitly for incorporating the transfer learning paradigm. More precisely, all three models are pre-trained on extensive unlabelled data first. Then, the adaptation to the fine-tuning setting is realized through an architectural change (explained in more detail in the following subsections) that is consistently applied to all three models (see Figure 15). However, it is important to note that due to the nature of static embeddings in the node2vec model used in the KG baseline, the KG baseline is rather a feature extraction than a transfer learning-based model architecture. Still, due to consistency reasons, it will be referred to as a transfer learning setting in the following.

In the following, Section 3.2.1 covers the NLP-baseline based on the BioBERT model, which only operates on text data. Then, Section 3.2.2 continues to explain the KG-baseline, which leverages node2vec to process sequential data generated from the triples in INDRA's text-triple pairs. Lastly, Section 3.2.3 introduces the proposed STonKGs model architecture, i.e., a multimodal Transformer that operates on combined input sequences of text and KG data extracted from the text-triple pairs in the INDRA dataset.

3.2.1 NLP Baseline

As previously mentioned, the NLP-baseline is based on the pre-trained BioBERT v1.1 model [Lee+20]. BioBERT v1.1 is a Transformer-based language model trained for 1 million steps on chunks of 512 tokens coming from a 4.5 billion token corpus extracted from PubMed abstracts (see [Lee+20] for more details on further hyperparameters). Therefore, rather than pre-training a new LM on the text evidences from INDRA's text-triple pairs from scratch, this thesis leverages the already pre-trained representations learned by the BioBERT model. There are two main reasons for that. First, this thesis is subject to limited computing power that can be utilized for the proposed experiments (see Section 3.4). Instead of reinventing the wheel for building purely text-based Transformers and spending weeks of available GPU resources on a pre-training procedure on text evidences from INDRA, it is arguably more expedient to use the available resources to explore the available resources pre-training procedures for the novel STonKGs model architecture.

Secondly, although the text corpus used for BioBERT is not entirely identical to the set of text evidences included in the text-triple pairs of the INDRA dataset, the two text sources are expected to have considerable overlaps since both are based on PubMed. The two major differences that are important to point out are the smaller size of the text evidence corpus in INDRA (i.e., roughly 13 million evidences with 50 tokens per evidence on average, resulting in $13 * 50 = 650$ million tokens, compared to 4,500 million tokens used in BioBERT) and the shorter input sequence length in INDRA (i.e., 50 tokens on average, compared to chunks of 512 tokens in BioBERT). Nonetheless, the practical implications imposed by the limited computing power, together with the overall similarity between the text sources used in BioBERT and INDRA, speak in favour of using the pre-trained BioBERT model over a new LM trained from scratch.

In order to apply the pre-trained BioBERT model to INDRA statements in later fine-tuning evaluation procedures, the contiguous string of text coming from the text evidence of each text-triple pair is first tokenized with the BioBERT tokenizer. Similar to the original BERT approach (see [Dev+19] for more details), the resulting input sequence is augmented with the special classification and separator tokens (i.e, `[CLS]` and `[SEP]`) and padded or truncated (padded in most cases, cf. Figure 14) accordingly to the expected input length of 512 tokens in BioBERT. Passing the input sequence through BioBERT yields a contextualized embedding sequence as its output (see the blue box in Figure 15). Each token embedding is represented as a 768-dimensional vector formed as a weighted average of its surrounding tokens (weighted by the attention coefficients) in this output sequence.

To adapt the pre-trained BioBERT model to the fine-tuning classification tasks (more details on the chosen tasks are provided in Section 3.3), additional model components, namely a pooling layer and a linear layer, were added to further process the output embedding sequence coming from the pre-trained BioBERT model (see the lower part of Figure 15). More specifically, pooling consists of using the special classification (i.e., `[CLS]`) token as an aggregated representation for the entire output sequence, analogous to the original BERT model [Dev+19]. Afterwards, the linear layer projects the pooled sequence onto class probabilities for the given fine-tuning task through a softmax activation function. While training the adapted model architecture on the task-specific data, all model weights are fine-tuned, including the weights of the BioBERT model.

3.2.2 KG Baseline

The starting point for the KG-baseline is a node2vec model [GL16] that is trained on the pre-training partition of the INDRA KG specified in Section 3.1.1. Excluding the triples that are later used in the fine-tuning datasets from the set of triples used for learning KG embeddings in the KG-baseline is of central importance. Otherwise, there would be information leakage (regarding the known connections of nodes in the KG) between the pre-training and fine-tuning parts of the overall transfer learning procedure (see Section 3.3 for more details on the pre-training and fine-tuning splits of the INDRA dataset). The node2vec model is trained for one iteration[8], using a random walk length of $l = 127$ and an embedding dimension size of $d = 768$ (the reasons for the choice of these values are given in Section 3.2.3). Both p and q are set to 1. Based on the set of random walks, word2vec is trained for four epochs to learn the embedding vectors for each node. In more detail, the internal word2vec model uses a window size of $w = 3$, $n = 5$ negative samples and a batch size of $b = 10,000$. Moreover, the word2vec model is optimized using the Stochastic Gradient Descent (SGD) optimizer [KW52] with a linearly decreasing learning rate starting at $lr = 0.025$.

Although a hyperparameter search (e.g., to find the optimal values of p and q) could have possibly helped to improve the later evaluation performance, it was not computationally feasible to conduct such a Hyperparameter Optimization (HPO) procedure given the size of the INDRA KG. Lastly, it should be pointed out that node2vec is a static embedding approach, contrary to the NLP-baseline covered in the previous subsection.

[8] It is important to note that the hyperparameter options offered by the framework used for the implementation of node2vec (see Section 3.4) are not entirely consistent with the notation and steps outlined in the original node2vec publication (see Algorithm 1).

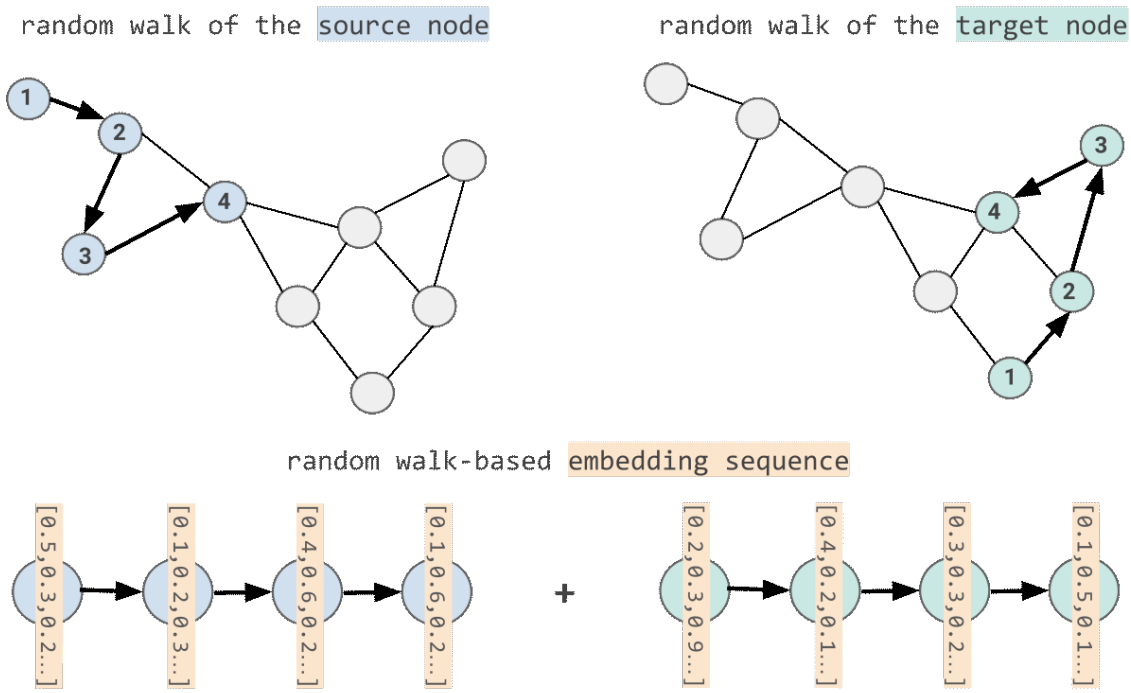


Figure 16: Generation of a random walk-based embedding sequence based on the source and target (i.e., head and tail) nodes of a given triple (the nodes labelled as number **1** in the left and right halves of the figure, respectively). The two random walks (shown by the bold black arrows) associated with the two nodes by the pre-trained node2vec model are used as a starting point. Then, for each node in the two random walks, the embedding vectors learned by node2vec (shown in orange) are accessed and concatenated into the final random walk-based embedding sequence (demonstrated in the lower part of the figure). (Image source: own.)

Therefore, the learned embeddings consist of a static lookup table that cannot be adapted in a fine-tuning procedure extended to previously unseen nodes. Hence, the initial output generated by the pre-trained node2vec model for an input triple of a given text-triple pair from INDRA consists of two embedding vectors of the source and target node of the triple (as node2vec is indifferent to relation types and thus unable to produce relation-specific embeddings). These embedding vectors implicitly contain the encoded context of the graph neighborhood structure.

However, the main incentive is to create a KG-baseline that can be used under comparable conditions with the NLP-baseline as well as STonKGs, which both operate on sequential input and output sequences. As a consequence, it is favourable to create a KG-baseline that also produces sequential outputs based on the two nodes in a given triple. This is where the random walks leveraged by node2vec come into effect. The overall goal is to create a so-called *random walk-based embedding sequence* $e(h_i, t_i)$ based on the source node h and target node t in a given triple (h_i, r_i, t_i) (see Figure 16). The

generation of the random-walk embedding sequence consists of three steps. First, the associated random walks $h = (h_i, ..., h_n)$ and $t = (t_i, ..., t_n)$ learned by the pre-trained node2vec model are extracted, starting from h and t, respectively. Secondly, each node n in the two walks is replaced by its embedding vector $\vec{e_n}$ learned by node2vec, resulting in two embedding sequences $(\vec{e_{h_i}}, ..., \vec{e_{h_n}})$ and $(\vec{e_{t_i}}, ..., \vec{e_{t_n}})$. Lastly, the two embedding sequences are concatenated to provide the overall random walk-based embedding sequence: $e(h_i, t_i) = (\vec{e_{h_i}}, ..., \vec{e_{h_n}}, \vec{e_{h_i}}, ..., \vec{e_{t_n}})$.

Analogous to Section 3.2.1, the same components used for the NLP-baseline are added to the KG-baseline to adapt the pre-trained model to the fine-tuning classification settings (see Figure 15). More specifically, each triple's random walk-based embedding sequence is pooled and passed through a linear layer to project the embedding dimension to the number of final classification labels. This model performs pooling using the dimension-wise maximum of the sequence embeddings to map the sequence to a single aggregated representation. As a result, and contrary to fine-tuning all the weights of the entire model architecture, the weights of the linear layer are the only additional model weights learned in the fine-tuning procedure of the KG-baseline.

3.2.3 STonKGs: A Sophisticated Transformer trained on biomedical text and Knowledge Graphs

The proposed STonKGs model architecture directly builds upon the NLP- and KG-baselines (referred to as NLP- and KG-backbones in the following) discussed in the previous subsections. Both the token embedding sequences produced by BioBERT [Lee+20] as well as the random walk-based triple embedding sequences constructed based on node2vec [GL16] are used to represent text and KG data in a shared model, respectively. More specifically, as shown in Figure 17, STonKGs is a multimodal single stream Transformer that uses a concatenation of these text- and KG-based embedding sequences as input to a cross encoder (similar to approaches like CoLAKE [Sun+20] and MDETR [Kam+21], which are used as inspiration for the methodology developed in this thesis).

The intuition behind this architectural design choice is that by passing a multimodal input sequence through a single stream Transformer, the model might be able to learn complex interdependencies between text and KG inputs. More specifically, since the output embedding representations returned by the Transformer are each weighted averages of its surrounding inputs, the Transformer is potentially able to incorporate context from the KG into the representation of a text token and vice versa. Figure 17 depicts the attention mechanism of the cross encoder, in which the attention coefficients are indicated

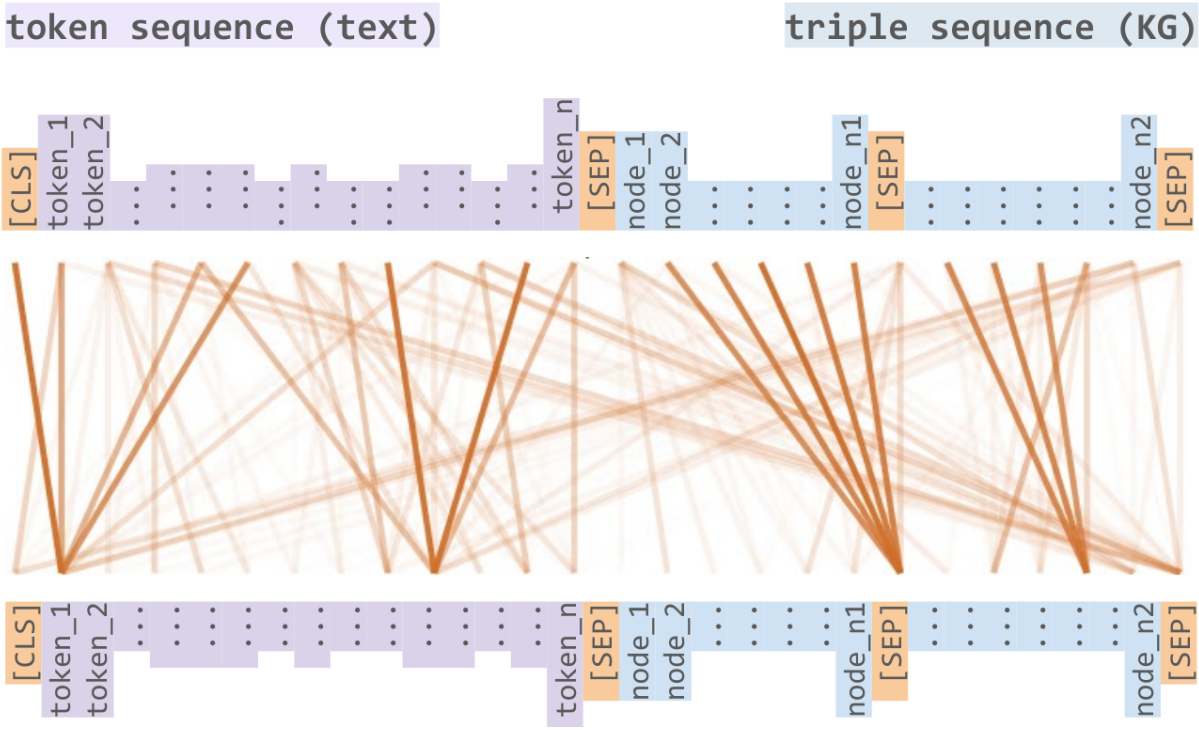


Figure 17: Input and output embedding sequences of the cross encoder used in STonKGs. This multimodal approach leverages the initial token embedding sequence produced by BioBERT (shown in purple) and the random walk-based triple embedding sequence generated based on node2vec pre-trained on the INDRA KG (shown in blue). The concatenation of the two embedding sequences is augmented with special classification and separator tokens to form the final input embedding sequence, which is subsequently processed by an attention mechanism (visualized by the orange links). (Image source: modified based on [Vas+17].)

by the orange links between the input and output sequences. With this approach, it is hoped to learn attention coefficients that can indicate which parts of an input (sub)graph or sentence are relevant to form the representation of a given unit of the overall multimodal input sequence[9].

In more detail, the STonKGs model architecture is a modification of the $\text{BERT}_{\text{BASE}}$ model [Dev+19] using the same maximum sequence length ($m = 512$), embedding dimension ($d = 768$), number of Transformer layers ($L = 12$) and number of attention heads ($A = 12$). The maximum sequence length is split in half to provide 256 tokens for text- and triple-based embedding sequences from INDRA's text-triple pairs and processed by the NLP- and KG-backbone, respectively. Moreover, similar to the original BERT model, there are several special tokens, such as the classification (`[CLS]`), separator (`[SEP]`)

[9] Whether the model is actually able to learn meaningful multimodal contexts or not would need to be further investigated (e.g., by taking a closer look at the values of the attention weights for a set of text-triple pairs). However, this aspect is beyond the scope of this thesis, as it is not central to proving or disproving the initial hypothesis.

and masking ([MASK]) tokens. Based on the combined input sequence generated by the backbones, the [CLS] token is added to the very beginning first, which is crucial for forming an aggregated representation of the input sequence later on. Next, three [SEP] tokens are added i) between the text- and triple-based input, ii) between the two random walks and iii) at the very end of the sequence (see Figure 17). These [SEP] tokens intend to i) structurally differentiate the text- and triple-based inputs as well as ii) the random walks from each other iii) and indicate the end of the entire sequence, similar to the original BERT model [Dev+19]. Lastly, the [MASK] token is used to mark the masked tokens in the pre-training objectives (described below). Analogous to BERT, positional and segment embeddings (also used to indicate the difference between text and KG input) are employed by STonKGs as well.

In the overall pre-training procedure applied to the text-triple pairs coming from the pre-training partition of the INDRA dataset (see Section 3.3), there are three training objectives. These training objectives are inspired by the original BERT model and applied to different subsets of the overall input sequence. All three objectives are jointly used to train the cross encoder of STonKGs:

1. **Masked Language Modeling (MLM)**: The MLM objective is adopted from the original $\text{BERT}_{\text{BASE}}$ model and applied to the text-based part of the multimodal input sequence (i.e., the first 256 tokens). Overall, the goal is to correctly predict specific text tokens that have been replaced by the special [MASK] token beforehand. This is realized through adding a so-called MLM head consisting of a linear layer and followed by a softmax function, which projects the output embedding sequences generated by the cross encoder to a vector of the size of the vocabulary used by the NLP-backbone (i.e., 30,000 tokens [Lee+20]). As a result, each dimension of an output produced by the MLM head represents one token of the overall vocabulary. The resulting values (normalized by the softmax function) in each dimension are the probabilities for a given masked token being the respectively represented token. Therefore, the true labels used in the MLM task consist of one-hot encoded vectors that contain the value 1 at the dimension of the underlying true token behind the [MASK] representation. STonKGs masks the same proportion of tokens as $\text{BERT}_{\text{BASE}}$, namely 15% of all tokens are chosen for replacement. From these tokens, 80% are masked, 10% are left unchanged, and the remaining 10% are replaced by a random token (to add further noise to the model, aiming at making the learning procedure more robust to errors) [Dev+19].

2. **Masked Entity Modeling (MEM)**: As the name suggests, this training objective

is derived from the MLM task and adapted to the triple-based part of the multimodal input sequence (i.e., the last 256 tokens). Here, the goal is to predict masked nodes (rather than masked text tokens) in the random walk-based sequence. The resulting MEM head again consists of a linear layer followed by a softmax. However, the output embedding representations are mapped to the "vocabulary size" (i.e., the number of unique nodes) of the INDRA KG used for pre-training (extended by the special `[SEP]` token to ensure consistency with the additional separators). Again, the underlying true labels for the masked nodes consists of one-hot encoded vectors that contain the value 1 at the index of the correct node behind the masked representation. It is important to note that the indices used for the nodes in the MEM head are entirely independent of the indices used for tokens in the MLM head since the two heads operate independently on different subsets of the combined input sequence. Only the initial masking operation is applied to the entire sequence before passing it through the cross encoder since the chosen proportions of masked tokens are kept entirely the same as in the previous training objective, which evidently speaks for a joint masking procedure.

3. **Next "Sentence" Prediction** (NSP): This training objective is also adopted from the original BERT model used as a basis for this multimodal Transformer. However, there is one major difference: Instead of predicting whether two sentences belong to each other or not, the NSP objective used in STonKGs is designed to distinguish valid text-triple pairs from randomly combined triples and text evidences. More specifically, this is achieved by augmenting the original pre-training dataset by a proportion of mismatching (i.e., negative) text-triple pairs. The chosen ratio of negative examples is 25%, which is lower than the 50% used in BERT. Nonetheless, the smaller proportion is selected on purpose, as it still incorporates the NSP objective without increasing the overall dataset size too much (ensuring that it is still computationally feasible to train the model for multiple epochs).

All in all, the losses of the three presented training objectives are added up to form the final loss function that is ought to be minimized during pre-training: $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{MLM}} + \mathcal{L}_{\text{MEM}} + \mathcal{L}_{\text{NSP}}$. STonKGs is pre-trained for 300,000 steps using a batch size of $b = 512$ and an Adaptive Moment Estimation with Decoupled Weight Decay (AdamW) [LH19] optimizer with a linearly decreasing learning rate starting at $lr = 10^{-4}$.

Following the same procedure outlined in the previous two subsections, STonKGs can be once again adapted to fine-tuning tasks operating on text-triple pairs using a classification component that consists of a pooling step and a linear layer (see Figure 15). In more detail,

pooling is performed using the `[CLS]` token embedding vector. During fine-tuning, all model weights, including the ones of the pre-trained cross encoder, are adapted based on the task-specific loss function.

3.3 Evaluation

As outlined in Section 1.2, the main incentive of this thesis is to test whether a model using both text and KG data (i.e., STonKGs) can lead to improved performance compared to two baselines (i.e., the NLP- and KG-baselines) that only use either one of the modalities. However, no clear definition has been given in this thesis so far regarding how this improved performance would need to look like. That is why this section provides an overview of the employed evaluation setting and the metrics used to define the model performances.

Generally, the idea behind the chosen evaluation setting for this thesis consists of applying the transfer learning paradigm (see [Goo+16, pp. 524-539] for an in-depth explanation on transfer learning) to evaluate all three models in a comparable setting. This means that there are two types of training procedures for each model: pre-training and fine-tuning. In the pre-training procedures employed in this thesis, the respective objectives are used to learn general task-independent (embedding) representations of text and (or) KG data in an unsupervised manner. The most common strategy to evaluate the pre-training procedure is to monitor the loss in regular intervals over all training steps. As the main focus of this thesis is to validate the novel STonKGs model architecture (rather than to verify the widely used node2vec and BioBERT models), the loss will only be recorded for the pre-training procedure of the STonKGs model.

The main focus of the evaluation procedure, however, lies in the fine-tuning tasks. Contrary to the pre-training procedure, the fine-tuning tasks consist of labelled datasets. These labelled datasets are used to measure the performance (occasionally referred to as the downstream performance in this thesis) of a model adapted to a specific task. Based on such a pre-trained model, this process can be repeated several times on different task-specific datasets, and the resulting set of tasks can be grouped into a benchmark. Hence, the evaluation procedure applied in this thesis consists of measuring the downstream performance of each of the three proposed models on each task in the constructed benchmark. All tasks described in the following are either binary or multiclass classification tasks to keep the tasks consistent and comparable to a reasonable degree. Each labelled task-specific dataset can be split into training and test splits that allow for fine-tuning the model parameters (as described in Section 3.2) using a loss function operating on the labelled data first. Then, the classification performance is evaluated on the test set

afterwards. All fine-tuning procedures of all three models are based on training for five epochs with a batch size of $b = 16$ on the respective training partition, using the AdamW optimizer and a linearly decreasing learning rate starting at $lr = 5 * 10^{-5}$.

First, the dataset splits used to separate the pre-training from the fine-tuning data are explained in more detail in Section 3.3.1. Next, Section 3.3.2 covers the specifics of the constructed benchmark used for evaluation, as well as of each fine-tuning task contained in it. Lastly, Section 3.3.3 is introducing another orthogonal evaluation aspect, namely ablation studies designed to measure the effect of certain model design choices employed in the STonKGs model architecture on the downstream performance.

3.3.1 Dataset Splits

Overall, the filtered set of INDRA statements (i.e., a set of text-triple pairs, of which some contain an annotation) is separated into two main disjunct components, namely the pre-training and fine-tuning splits. The main reason for this disjunct split is to prevent information leakage between the pre-training part and the evaluation of the models on the fine-tuning tasks. Otherwise, if the pre-training partition contained text-triple pairs that are also present in the fine-tuning datasets, the models could potentially exploit previously learned characteristics of the respective data entries. This might skew the downstream performance (i.e., the performance might look better compared to an evaluation on actual out-of-distribution (OOD) data) and contradicts the idea of an unbiased transfer learning procedure[10].

Since pre-training procedures aim to learn general representations that can be adapted to various use-cases, they typically require large datasets (such as a 3.3 billion word corpus for BERT [Dev+19], a 4.5 billion corpus for BioBERT [Lee+20] or even a 300 billion token corpus for GPT-3 [Bro+20]). Therefore, the majority of the 13,609,994 pre-filtered text-triple pairs were used for the pre-training partition. More specifically, since the pre-training procedure is unsupervised, all INDRA statements (i.e., text-triple pairs) with no additional annotation (see Section 3.1.3) are used as a basis for the creation of the pre-training dataset. Only two small fractions of the unannotated statements, containing 12,836 and 78,979 text-triple pairs, respectively, are filtered out beforehand, as they are

[10]Note that although this reasoning might appear to be evident, it is not consistently applied in many well-known publications. For instance, BERT [Dev+19] has been primarily trained on text data from Wikipedia, and several of the GLUE [Wan+18] tasks used for evaluation have been extracted from Wikipedia as well. However, Devlin et al. did not mention any explicit checks on possible information leakage.

used for two types of fine-tuning tasks that do not require annotations for the construction of the particular labels (explained in more detail in the next subsection).

With regards to the INDRA statements that contain some type of annotation ($n = 1,507,150$, see Section 3.1.3), the vast majority of annotations are disregarded (as they are not suitable for the creation of biologically relevant evaluation tasks) and grouped with the unannotated statements for pre-training. Only a small partition of INDRA statements that contain some relevant annotation and pass the evaluation task-specific filtering steps (35,334 text-triple pairs chosen from the significantly larger set of text-triple pairs including any of the four relevant annotation types, see Section 3.1.3) is used for the fine-tuning partition. In total, **127,149 text-triple pairs (0.93%)** are used for **all fine-tuning tasks**, leaving **13,482,845 text-triple pairs (99.07%)** for the **pre-training procedure** of STonKGs and the KG-baseline (i.e., to form the INDRA KG used for training node2vec).

3.3.2 Fine-tuning Tasks

As explained before, the general idea behind the fine-tuning setting is to evaluate the general representations learned by the three pre-trained models on multiple classification tasks that represent a wide range of biologically relevant use-cases. Each task-specific fine-tuning dataset is further split into training and test splits. While the training split adapts the pre-trained representations with classification-specific loss function, the test split is used to measure the models' performances on previously unseen data. In general, any biological application represented by labelled text-triple pairs can be used to construct a classification task to evaluate the three models. The only limitation imposed by the model designs is that the nodes contained in the triples need to be included in the INDRA KG because it is impossible to create embedding representations for previously unseen nodes in node2vec. In this thesis, however, subsets of the filtered INDRA dataset (i.e., the fine-tuning split described in the previous paragraph), based on certain special characteristics, were chosen as a starting point for constructing a total of eight classification tasks. These tasks can be grouped into three different biological application types (see Table 6). Examples for the text evidences in each of the eight tasks are provided in Table 7, and the class distributions in each task are shown in Figure 18.

In all eight tasks, multiple text-triple pairs based on the same text evidence (i.e., different triples that were extracted from the same text source) were defined as duplicates. Therefore, only one text-triple pair from each set of duplicates was kept. The main reason for this additional filtering step in the fine-tuning tasks is to avoid modality-specific biases:

Task type	Task name	Dataset size	# of classes	Description	Class labels	Ontology
Relation type	**1) Polarity**	78,979	Binary	Up-/down-regulation of the target caused by the source	Increase, decrease	-
	2) Interaction type	78,979	Binary	Physical or non-physical interaction between the source and target	Direct, indirect	-
Context annotation	**3) Cell line**	3,760	10	Cell line context	HEK293, DMS114, HeLa, NIH-3T3, HepG2, MCF7, COS-1, THP-1, LNCAP, U-937[11]	Cell Line Ontology (CLO) [Sar+14]
	4) Disease	4,586	10	Disease context	Neuroblastoma, breast cancer, lung cancer, atherosclerosis, multiple myeloma, leukemia, melanoma, osteosarcoma, lung non-small cell carcinoma	Disease Ontology (DO) [Sch+12]
	5) Location	5,223	5	Cellular location context	Cell nucleus, extracellular space, cell membrane, cytoplasm, extracellular matrix	Medical Subject Headings [Lip00]
	6) Species	21,765	3	Species context	Human, mouse, rat	NCBI Taxonomy [Sch+20b]
Annotation error	**7) Annotation error (binary))**	12,836	Binary	Whether the extracted triple is extracted correctly or not	Correct, incorrect	-
	8) Annotation error (multi-class)	12,611	8	Whether the extracted triple is extracted correctly or not (including all error types)	Correct, no relation, wrong relation, grounding, polarity, act vs amt, entity boundaries, hypothesis	-

Table 6: Overview of the eight classification tasks used to create the evaluation benchmark, grouped by their task types. For the context annotation tasks (task 3-6), the ontologies used by INDRA to ground the annotations used as class labels are listed as well. While the datasets used in each context annotation task are distinct from each other, the datasets used in task 1-2 as well as 7-8 are the same/largely overlapping.

Task name	Example
1) Polarity	"HSP70 [...] increases ENPP1 transcript and protein levels" [Mar+09]
2) Interaction type	"SHP repressed [...] transcription of PEPCK through direct interaction with C/EBPalpha protein" [Par+07]
3) Cell line	"We show that upon stimulation of HeLa cells by CXCL12, CXCR4 becomes tyrosine phosphorylated" [Cha+05]
4) Disease	"[...] nicotine [...] activates the MAPK signaling pathway in lung cancer" [Tro+04]
5) Location	"The activated MSK1 translocates to the nucleus and activates CREB [...]." [Dea+98]
6) Species	"Mutation of putative GRK phosphorylation sites in the cannabinoid receptor 1 (CB1R) confers resistance to cannabinoid tolerance and hypersensitivity to cannabinoids in mice" [Mor+14]
7) Correct/Incorrect (Binary)	Examples are available at INDRA's curation guidelines[12].
8) Correct/Incorrect (Multiclass)	

Table 7: Examples for the text evidences included in each of the eight fine-tuning tasks.

While the KG-baseline, as well as STonKGs, would recognize duplicate text-triple pairs as different entities (due to their distinct triple characteristics), the NLP-baseline would fail to do so, as it solely operates on text. If such duplicates were present across the training and test split of the fine-tuning tasks, there would be a high chance of the NLP-baseline exploiting statistical cues, as a duplicate used in the test set might have already been present in the training data.

First, two so-called **relation type** classification tasks are testing the models' abilities to differentiate between opposite relation types present in protein-protein interactions (PPIs). The first task, termed the **(1) polarity task**, deals with the binary classification of **n=78,979** text-triple pairs that describe either up- or down-regulated processes. An up-regulation is characterized by a biological entity that increases the quantity of another one, whereas a down-regulation describes a decreasing quantity. Hence, the classes are referred to as *increase* and *decrease* in the following. The second task, the **(2) interaction type** task, distinguishes between direct and indirect interactions in the same dataset that is used for the polarity task (this time, however, using the *direct* and *indirect* class labels). Peng et al. define the two interaction types as follows: "An interaction is 'direct' if the molecular interfaces of two proteins contact with each other; otherwise an interac-

[12] https://indra.readthedocs.io/en/latest/tutorials/html_curation.html#curation-guidelines (accessed on August 9th, 2021)

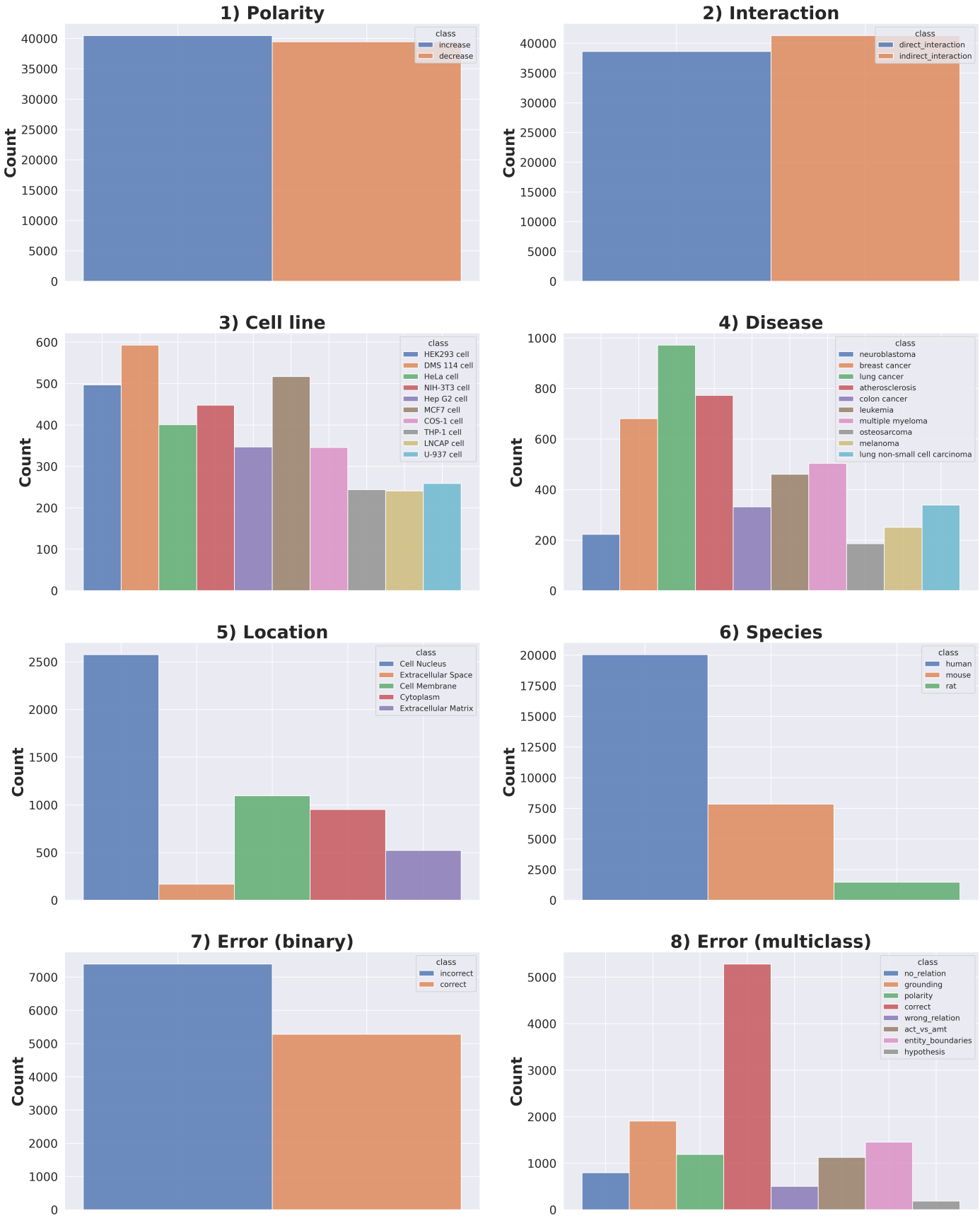


Figure 18: Class distributions for all eight fine-tuning tasks used in the evaluation benchmark. A basic degree of class balance was ensured by limiting the maximum number of classes to ten and leaving out classes present in less than 1% of the labels. However, there is still a considerable class imbalance in some tasks (i.e., the species, location and annotation error (multiclass) tasks). (Image source: own.)

tion is 'indirect' if two proteins are physically separated, but they interact through other intermediates and build a complex." [Pen+17]. The labels for both classification tasks are extracted from the intermediate BEL representation of the triples in the text-triple pairs. More specifically, four relation types are mapped to the binary labels for the two tasks (summarized in Table 8), respectively: direct increase (DIR-INC), direct decrease (DIR-DEC), indirect increase (INC) and indirect decrease (DEC). Overall, the two relation type tasks intend to test how well the nature of those relations is (implicitly) encoded in text and (or) the KG, which is particularly interesting for the models that include KG data (i.e., the KG-baseline and STonKGs) since node2vec does not explicitly include the triples' relation type.

Next, there are four multiclass **context annotation** tasks that evaluate whether the models are able to classify four different types of biological contexts, namely the **(3) cell line**, **(4) disease**, **(5) location** and **(6) species** in which a given biological interaction described by a text-triple pair can occur in. All of the four classification tasks are directly based on the annotations provided in the INDRA statements. However, since the annotations in the INDRA statements are based on a set of ontologies, there is a vast number of possible values for each context type (e.g., more than 250,000 possible species names based on the NCBI taxonomy [Sch+20b]). Moreover, the distribution across this vast number of classes is tremendously skewed, as the majority of the available biological literature is concentrated on certain fixed experimental settings (for instance, drugs are mainly tested on humans, mice or rats rather than birds). To construct appropriate multiclass classification tasks with a reasonable number of classes and class distribution, two constraints were applied to the unfiltered datasets: i) A maximum of ten classes is allowed in each of the four tasks, and ii) each possible class needs to be represented by at least 1% of the total number of labels. As a consequence, the number of classes in each of the four tasks ranges from three to ten classes (see Table 6 for a complete list of all possible classes for each task). Moreover, the resulting datasets sizes are: **n=3,760 (cell line)**, **n=4,586 (disease)**, **n=5,223 (location)**, **n=21,765 (species)**. Altogether, the context classification tasks are specifically relevant in drug discovery and clinical applications. In such applications, certain biological effects might be different, if not even entirely reversed, depending on the species, disease, cell line or cellular location they are observed in.

Lastly, there are two **annotation error**-based tasks, which aim at providing a meta-level

[12]The full names of the cell line classes are omitted due to the lack of space. See `https://www.ebi.ac.uk/ols/ontologies/clo` (accessed on August 9th, 2021) for the full names and further descriptions.

Relation	1) Polarity	2) Interaction type
Increases (INC)	Increase	Indirect
Decreases (DEC)	Decrease	Indirect
Directly Increases (DIR-INC)	Increase	Direct
Directly Decreases (DIR-DEC)	Decrease	Direct

Table 8: Mapping between the BEL relation types (listed in the leftmost column) and the labels for the two relation-type classification tasks. As a result, the same dataset can be used to construct both the polarity and the interaction type tasks.

perspective on the quality control aspect of the INDRA dataset. Since the majority of the extracted text-triple pairs are stemming from automated reading/text mining systems (see Table 1), certain types of processing errors cannot be entirely ruled out. For instance, two common error types are the extraction of the wrong relation and the grounding of nodes to the wrong normalized terms in a given ontology. Linking the presence of errors (or even the specific error types) to certain characteristics in the text-triple pairs can potentially help to identify reasons for those annotation errors. Hence, if the proposed STonKGs model can outperform the two baselines on this task type, it could potentially be employed in quality control pipelines in the future. To test the models' abilities to detect erroneous text-triple pairs, two classification tasks are constructed: The **(7) annotation error (binary)** and the **(8) annotation error (multiclass)** tasks. Based on a manually annotated dataset from the INDRA labs team at HMS, the respective binary or multiclass class labels are created by either grouping all error types together and discriminating between erroneous text-triples and correctly extracted ones (resulting in the *incorrect* and *correct* for the binary task), or by treating each error type as well as the correctly extracted cases as an individual class label (resulting in a total of eight labels, as listed in Table 6). Some error types make up less than 1% of the overall labels. Therefore they were filtered out. As a result, the dataset used for the annotation error (multiclass) task is slightly smaller (**n=12,611**) than the one used for the binary task (**n=12,836**).

In each of the eight classification tasks, the fine-tuning model architecture adaptations described in Section 3.2 are used to generate the respective class probabilities, and the class with the highest probability is chosen as the predicted label. During the fine-tuning procedure, all pre-trained model weights are further trained using binary or multiclass cross-entropy loss functions (see Formula 11 and 12, based on [Con19]) to compare the predicted labels $\hat{y}_i$ to the true labels y_i for all data entries n. In the binary case, the ground truth labels y_i consist of either zero or one (using *increase*, *direct* and *correct* as the positive classes for task 1, 2, and 7, respectively). This idea is extended to the

multiclass case, in which the true label for the i-th data entry is only equal to one ($y_{ic} = 1$) if the class c is the correct class and is zero otherwise ($y_{ic} = 0$). Afterwards, the models are used to predict the labels for the test split, and F1-scores are employed to measure the test performance (see Formula 13 and 14, based on [Dev20]). More precisely, in the binary case, the F1-scores are calculated based on the number of true positives TP, false positives FP and false negatives FN included in the precision $P = \frac{TP}{TP+FP}$ and recall $R = \frac{TP}{TP+FN}$ scores. For the multiclass tasks, the F1-scores are first calculated for each class in a "one-versus-all" setting (resulting in class-specific values R_c and P_c for recall and precision, respectively). Then, the total F1-score is formed as a sum of all class-specific F1-scores, weighted by the relative number of entries for each class.

$$\mathcal{L}_{\text{binary}} = -\sum_{n=1}^{N} y_i \log \hat{y}_i + (1 - y_i) \log (1 - \hat{y}_i) \tag{11}$$

$$\mathcal{L}_{\text{multiclass}} = -\sum_{n=1}^{N} \sum_{c=1}^{C} y_{ic} \log \hat{y}_{ic} \tag{12}$$

$$\text{F1} = 2 * \frac{P * R}{P + R} \tag{13}$$

$$\text{F1} = \sum_{c=1}^{C} \frac{n_c}{N} * 2 * \frac{P_c * R_c}{P_c + R_c} \tag{14}$$

Overall, the training and testing process is repeated multiple times within the framework of a five-fold[13] cross-validation procedure with changing train-test splits. (However, the train-test splits are the same for all models). The final reported F1-scores are the means as well as the standard deviations across the five recorded folds. It is important to note that the standard deviations should be taken with a grain of salt, as they only cover the variance across a very low number (i.e., five) splits. Hence, the sample size is too small for quantitative statistical tests (e.g. a t-test or a Mann-Whitney U test). Instead, the differences in F1-scores will be described qualitatively. More specifically, to describe the differences, both the absolute and the relative improvements (see Formula 15) of the best STonKGs variant (explained in the following subsection) over the best baseline model on each fine-tuning task are reported in Section 4.

$$\text{F1-difference}_{\text{relative}} = \frac{\text{STonKGs}_{\text{best}} - \text{Baseline}_{\text{best}}}{\text{Baseline}_{\text{best}}} \tag{15}$$

[13] A larger number of folds would not have been computationally feasible, as it would have significantly increased the required runtime.

3.3.3 Ablation Studies

In addition to comparing the performances of the two baseline models and STonKGs, this thesis also intends to investigate the role of specific pre-training and model design choices on the performance of the STonKGs model itself. A common method for the examination of such effects is the conduct of so-called ablation studies. The goal of an ablation study is to leave out one specific model component and observe how the model performance changes. Based on a default model, the two aspects chosen for the ablation studies conducted in this thesis are changes in the number of training steps and the exclusion of the NSP objective. All ablations are applied to the pre-training procedure of STonKGs, and their effects are analyzed based on changes in the downstream performance on the evaluation benchmark. In total, there are three variants of STonKGs that are presented in Section 4:

1. **STonKGs$_{\mathbf{300k}}$ (default)**: This model variant is the basis for the following two ablated versions. More specifically, STonKGs$_{300k}$ is pre-trained for 300,000 training steps using the hyperparameters outlined in Section 3.2.3, including all three pre-training objectives (MLM, MEM and NSP). The reason for choosing precisely 300,000 steps is to reach a proportionate order of magnitude for the number of pre-training steps compared to the BioBERT model used as the NLP-baseline, given the available compute power and runtime. In more detail, BioBERT v1.1 has been pre-trained for 1 million steps using a batch size of $b = 192$ [Lee+20] (resulting in $192 * 10^6 = 192,000,000$ total training samples that are passed through BioBERT during pre-training), whereas STonKGs is pre-trained for $300,000$ steps using a batch size of $b = 512$ (resulting in $512 * 300,000 = 153,000,000$ total training samples). (Note that this is a very vague comparison since the average number of tokens per input sequence (before padding) is significantly different in BioBERT and STonKGs.)

2. **STonKGs$_{\mathbf{150k}}$**: The second variant is the realization of the ablation regarding the number of training steps. Instead of using the full 300,000 steps, STonKGs$_{150k}$ is only trained for half as many steps. This ablation intends to investigate whether fewer training steps significantly decrease the model performance or not. It should be noted that rather than pre-training a new model from scratch, this ablation is realized through model checkpointing (i.e., STonKGs$_{150k}$ is an interim checkpoint of STonKGs$_{300k}$).

3. **STonKGs$_{\mathbf{NO\ NSP}}$**: Since the effectiveness of the NSP objective has been questioned in existing literature before [Liu+19], the STonKGs$_{\text{NO NSP}}$ model variant is

constructed to measure the effect of the inclusion or exclusion of the NSP objective on the downstream model performance. Due to computational constraints, however, this model can only be trained for 150,000 instead of the full 300,000 steps. Hence, the effect of the NSP objective on the overall performance is ought to be analyzed by comparing $\text{STonKGs}_{\text{NO NSP}}$ to $\text{STonKGs}_{\text{150k}}$ rather than $\text{STonKGs}_{\text{300k}}$.

3.4 Implementation

All of the code used in this thesis is comprised in the `stonkgs` package[14] written in Python [VD95] (v3.8.8). The two main components of the `stonkgs` package are the `stonkgs.data` and `stonkgs.models` submodules that contain all relevant code for the data processing as well as the model creation and training steps. However, since the INDRA dataset is not publicly available, none of the raw data, the pre-training dataset or the fine-tuning evaluation datasets is included in the remote repository (instead, they only exist in local copies of the repository). Moreover, `stonkgs` also contains several Jupyter Notebooks [Klu+16] that are primarily used to generate data statistics.

Although a complete overview of all Python packages and frameworks used for implementation is beyond the scope of this thesis, several central packages are discussed in the following (summarized in Table 9, in which all links to the packages are listed as well). With regards to processing the INDRA statements (and creating a BEL graph), the implementation relied on the respective `indra` package as well as `pybel`. Next, the node2vec implementation from the `nodevectors` repository was used to train the node2vec model, and the KG-baseline was constructed using a `pytorch-lightning` module. The two Transformer-based models (i.e., the NLP-baseline as well as STonKGs) were implemented using HuggingFace's `transformers` library [Wol+20] with a `pytorch` [Pas+19] backend. The `transformers` package enabled to pre-train STonKGs in the half precision floating-point format (FP16), which significantly reduces the required computational resources for most operations (e.g., the weight updates). Another employed technique used to lower the required GPU memory is gradient accumulation. This method lowers the number of training examples that are passed to a GPU at once in two steps: First, the given effective batch size ($b = 512$ in STonKGs) is split into multiple parts (eight parts, in this case, resulting in a batch size of $b_{\text{acc}} = 64$) and the gradient for each part is calculated individually. In the second step, the weights are updated with the accumulated gradients. Overall, both the pre-trained process as well as the fine-tuning procedures are logged using the `mlflow` package, which tracks the pre-training loss, the evaluation

[14] The source code is publicly available at `https://github.com/stonkgs/stonkgs` (accessed on August 22nd, 2021).

Category	Python Package	Purpose	Documentation/Link	Version
Data	**indra**	Process INDRA statements	https://indra.readthedocs.io/en/latest/	1.19.0
	pybel	Create the KG from INDRA statements	https://pybel.readthedocs.io/en/latest/	0.15.2
Models	**nodevectors**	Train node2vec embeddings	https://github.com/VHRanger/nodevectors	0.1.23
	torch	Pytorch backend for all three models	https://pytorch.org/docs/stable/index.html	1.8.1
	pytorch-lightning	Implement the KG-baseline	https://pytorch-lightning.readthedocs.io/en/latest/	1.2.3
	transformers	Implement the NLP-baseline and STonKGs and create model checkpoints	https://huggingface.co/transformers/	4.6.1
Other	**mlflow**	Logging all results	https://www.mlflow.org/docs/latest/index.html	1.15.0
	sklearn	Calculate the metrics	https://scikit-learn.org/stable/modules/classes.html	0.24.1

Table 9: Overview of the most important Python packages, their categorization, purpose and the employed version used in the implementation of this thesis. A complete list of all packages used in the source code of this thesis is provided in the `setup.cfg` file included in the `stonkgs` package. All listed web pages were accessed on August 5th 2021.

metrics (calculated using `sklearn`) as well as a variety of hyperparameters. Moreover, during pre-training, model checkpoints are created in regular intervals (i.e., every 2500 training steps) since the pre-training procedure needs to be interrupted and continued without losing the progress.

In more detail, the NLP-baseline is initialized by loading the pre-existing *dmis-lab/biobert-v1.1* model uploaded by Lee et al. on the HuggingFace model hub[15]. The main in-

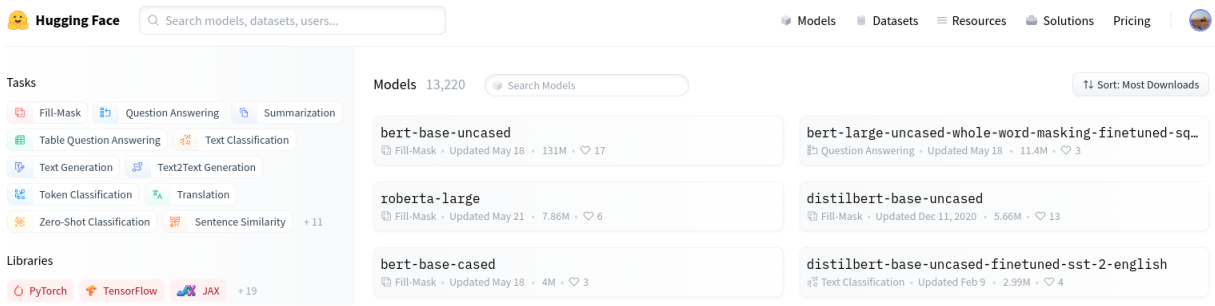
[15] https://huggingface.co/dmis-lab/biobert-v1.1 (accessed on August 5th, 2021)

centive of the HuggingFace model hub platform (shown in Figure 19a) is to provide the possibility for researchers to share pre-trained models with the NLP community so that they can be easily re-used by other users and integrated into their use-cases with the existing `transformers` Application Programming Interface (API). In particular, this is convenient for users that do not have access to powerful computer clusters that are needed to pre-train most Transformer-based models, since they can simply download and re-use already pre-trained models. Contrary to the NLP-baseline, STonKGs is implemented with custom written Python classes (`STonKGsForPreTraining` and `STonKGsForSequenceClassification`) that mimic the same behavior as the two respective classes from the `transformers` package (achieved through class inheritance from the `BertForPreTraining` as well as the `BertForSequenceClassification` classes). The main reason for choosing to inherit from these two classes is that it enables to share the pre-trained STonKGs models (i.e., STonKGs$_{300k}$ and STonKGs$_{150k}$) in a straightforward manner on the model hub as well[16] (see Figure 19b). More specifically, the uploaded STonKGs models allow future users to utilize the `stonkgs` package and download the already pre-trained models through the default `from_pretrained` function that is included in all model classes in the `transformers` package, as well as the STonKGs-specific classes that inherit from them.

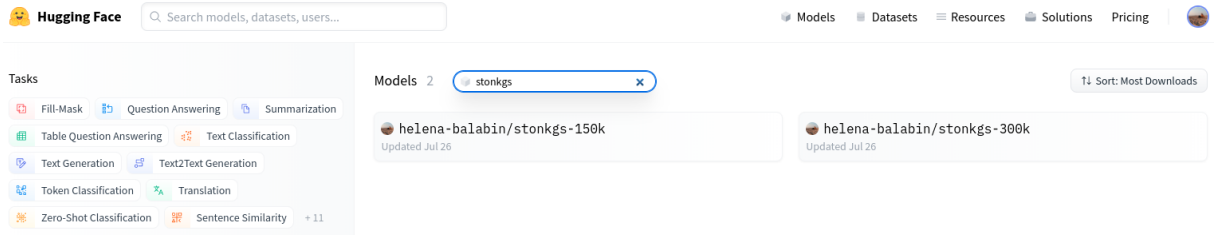
With regards to the computational resources used in this thesis, most of the required training procedures are GPU- rather than CPU-intensive. The only exception to this is the pre-training procedure of the KG-baseline, for which the main limitation is not imposed by the demanded GPU power but by the required Random-Access Memory (RAM) since the entire KG needs to be loaded and processed in memory at once. Hence, the KG-baseline is the only model that has been trained on a symmetric multiprocessing (SMP) compute cluster node with four Intel Xeon Platinum 8160 processors and 1.5 terabytes RAM. All other models are trained and evaluated on another compute cluster node with four NVIDIA A100 Tensor Core GPUs with 40 gigabytes of memory each (a single-machine multi-GPU node referred to as the GPU node in the following).

The major bottleneck for the computational tasks executed in this thesis is the available runtime on the GPU node, limited by i) the six-month duration of this thesis as well as ii) deadlines that need to be met by other users that require the GPU node. As a result, a total of roughly one and a half months of runtime on the GPU node can be seen as

[16] STonKGs$_{300k}$ and STonKGs$_{150k}$ are available at `https://huggingface.co/stonkgs/stonkgs-300k` and `https://huggingface.co/stonkgs/stonkgs-150k`, respectively (accessed on August 9th, 2021).



(a) Most popular models in the HuggingFace model hub.



(b) Searching for the STonKGs models in the HuggingFace model hub.

Figure 19: Screenshot of the HuggingFace model hub, which is a platform used to share pre-trained Transformers with the NLP community. (Image source: taken from [Con21].)

a realistic estimate to conduct all experiments. Hence, the main factors influencing the runtime have been adapted accordingly. These factors include the number of pre-training steps for STonKGs or the number of cross-validation folds regarding the repetition of the evaluation procedure on the fine-tuning tasks. The vast majority of the available GPU was utilized for the pre-training procedure of STonKGs. More specifically, it took 284.18 hours (≈11.84 days) and 568.35h hours (≈23.68 days) to pre-train the STonKGs$_{150k}$ and STonKGs$_{300k}$ models, respectively (note that the pre-training time of STonKGs$_{150k}$ is included in the time listed for STonKGs$_{300k}$). Moreover, the pre-training procedure for STonKGs$_{\text{NO NSP}}$ took additional 289.73 hours (≈12.07 days), as it needed to be pre-trained from scratch.

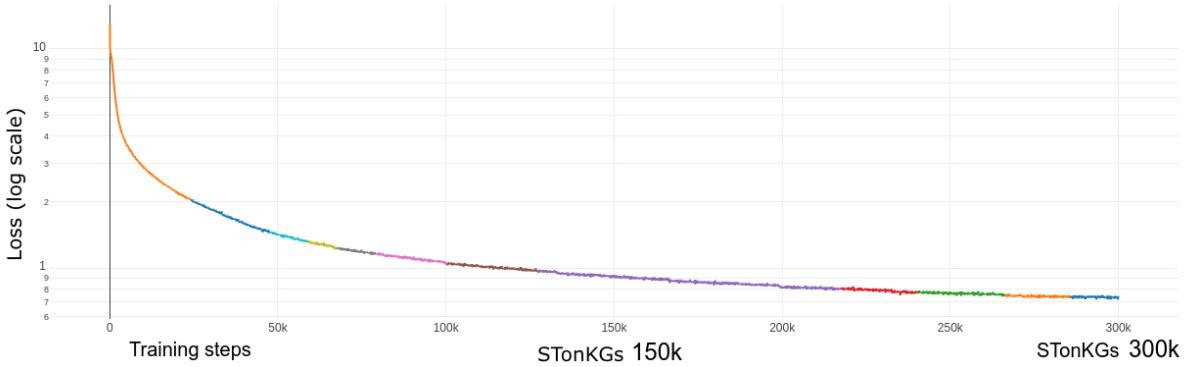
4 Results

Similar to the clear distinction of the pre-training and fine-tuning parts in Section 3, the main results presented in this section can be grouped into these two categories once again. With regards to pre-training, the main focus lies on the novel STonKGs model rather than the two baseline models. As outlined in Section 3.3, the loss of the joint training objective used in STonKGs is the chosen metric used to assess the progress of the pre-training procedure. That is why in Section 4.1, the loss curves of STonKGs_{150k}/STonKGs_{300k} as well as $\text{STonKGs}_{\text{NO NSP}}$ are presented and analyzed in more detail. However, the centerpiece of this section consists of the fine-tuning performances described in Section 4.2, based on the benchmark model performances reported in Table 10. This central table lists a total of five models (i.e., two baseline models and the three STonKGs variants) evaluated on each of the eight classification tasks, resulting in 40 reported performances. More specifically, Table 10 provides the basis for an in-depth analysis of performances from three different perspectives, namely based on the differences between STonKGs and the baselines (Section 4.2.1), between STonKGs and its ablations (Section 4.2.2) as well as across the different tasks (Section 4.2.3). Furthermore, Table 10 also includes the absolute and relative differences in performance between the best baseline and the best STonKGs variant. All in all, this section intends to provide an all-encompassing view on the comparison of all proposed models on all constructed tasks, which serves as a basis for the interpretation of the juxtaposition of models provided in Section 5.

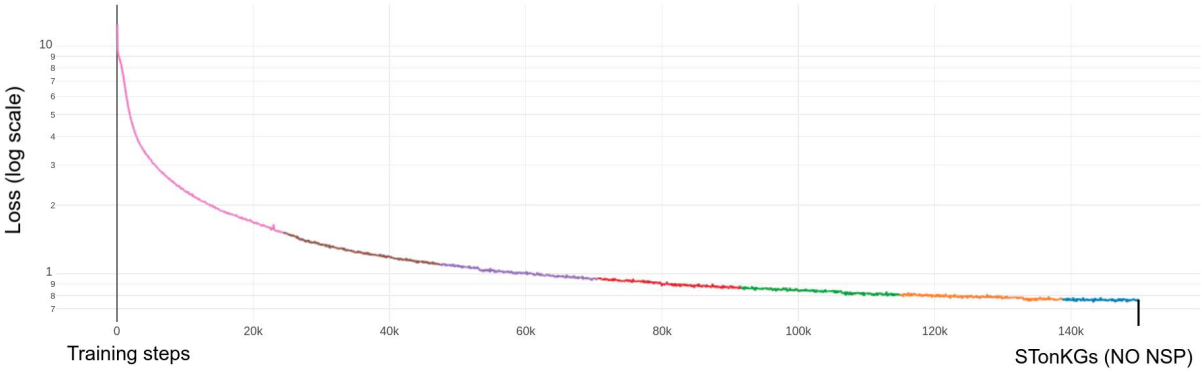
4.1 Pre-Training STonKGs

In Figure 20, the loss curves (i.e., the value of the loss as a function of the number of training steps) for all three STonKGs variants are shown. It should be emphasized that the loss curve of STonKGs_{150k} is a subset of the loss curve of STonKGs_{300k} (since STonKGs_{150k} is implemented through an interim checkpoint of the pre-training procedure of STonKGs_{300k} at 150,000 steps in order to avoid another redundant and computationally expensive pre-training procedure). Therefore, the losses of STonKGs_{150k} and STonKGs_{300k} are represented by the same loss curve (see Figure 20a). On the other hand, $\text{STonKGs}_{\text{NO NSP}}$ requires a pre-training procedure from scratch, since its join loss (i.e., $\mathcal{L}_{\text{NO NSP}} = \mathcal{L}_{\text{MLM}} + \mathcal{L}_{\text{MEM}}$) is different from the loss used for the other two STonKGs variants (i.e., $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{MLM}} + \mathcal{L}_{\text{MEM}} + \mathcal{L}_{\text{NSP}}$). Moreover, it should be noted that the pre-training procedure requires large amounts of compute power resulting in a long runtime (see Section 3.4), and hence needs to be conducted in many separate parts[17], indicated by the differently colored segments of the loss curve shown in Figure 20.

[17] Mainly caused by the maximum 48-hour runtime limit on the GPU compute node used in this thesis.



(a) Loss curve of the joint pre-training procedure of STonKGs_{150k} and STonKGs_{300k} (i.e., STonKGs_{150k} is an interim checkpoint of STonKGs_{300k}, as indicated on the x-axis), based on the $\mathcal{L}_{\text{total}}$ loss, consisting of the MLM, MEM and NSP pre-training objectives.

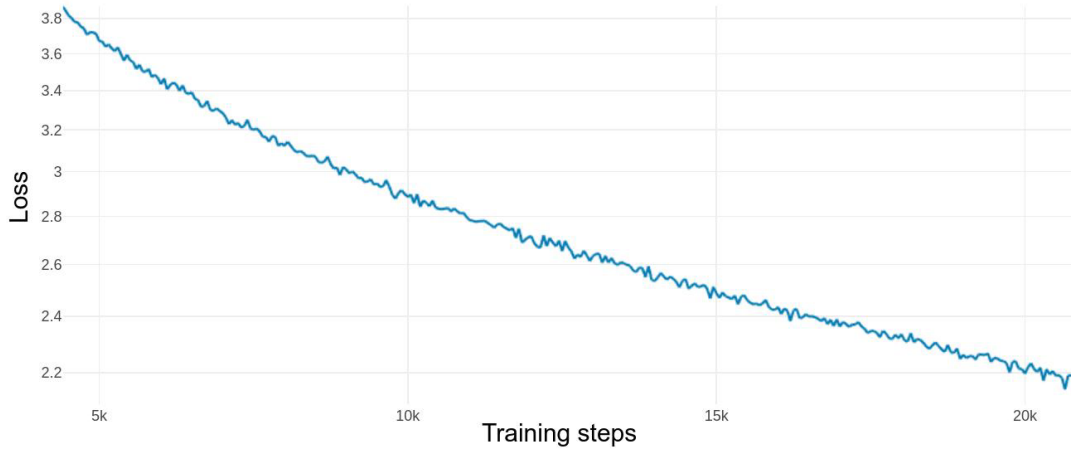


(b) Loss curve of the pre-training of $\text{STonKGs}_{\text{NO NSP}}$, based on the $\mathcal{L}_{\text{NO NSP}}$ loss, consisting of the MLM and MEM pre-training objectives.

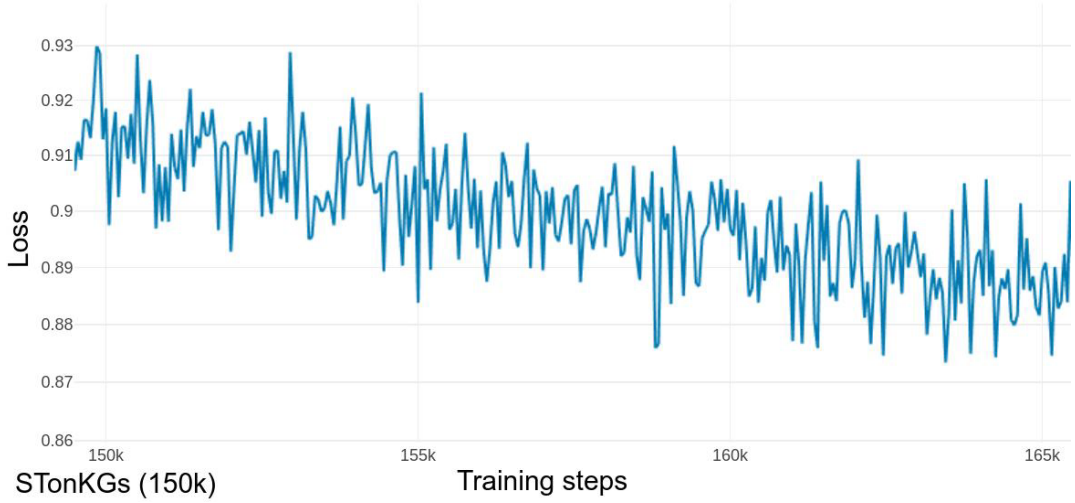
Figure 20: Pre-training loss curves of the different STonKGs variants for all 300,000 training steps. While Figure 20a is showing the loss curve of STonKGs_{150k} and STonKGs_{300k}, Figure 20b is displaying the loss curve of the $\text{STonKGs}_{\text{NO NSP}}$ model. In both parts, the x-axis is depicting the number of training steps at which the loss was captured (note that the shown intervals are different in 20a and 20b), whereas the y-axis is showing the value of the loss on a logarithmic scale. In both pre-training procedures, the loss was recorded every 50 steps, resulting in 6,000 and 3,000 data points in Figure 20a and 20b, respectively. The different colors appearing in the loss curves show the different runs (i.e., interruptions and re-starts of the pre-training procedures) in which the models were sequentially pre-trained. (Image source: own.)

The first main general observation regarding the pre-training loss of STonKGs is that the loss continuously decreased with an increasing number of training steps for all model variants. Furthermore, the main decrease of the loss can be seen in the first parts of both pre-training procedures. Afterwards, the loss curves become increasingly flat (but continue to decrease slightly). For instance, while pre-training STonKGs_{300k}, more than 94% of the loss reduction is taking place within the first 50,000 training steps, whereas the following 250,000 training steps merely account for the remaining 6%[18]. Additionally, the

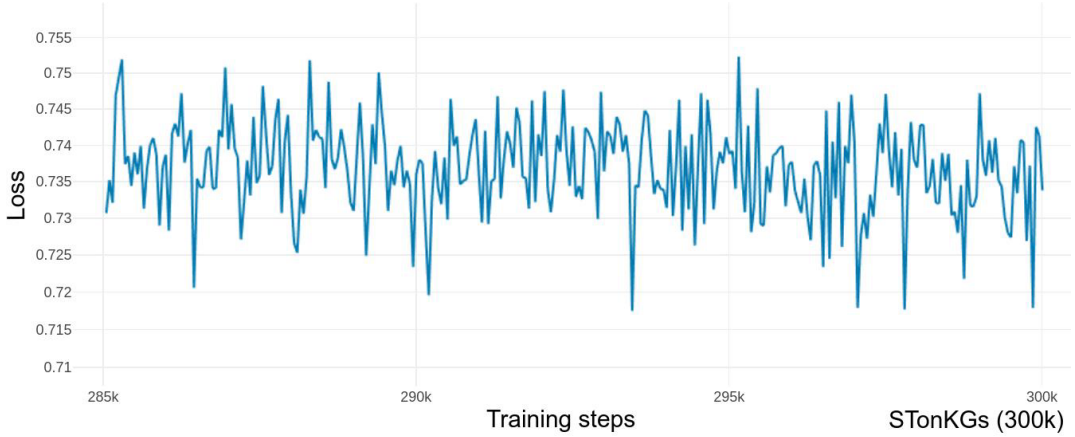
[18]The values of the respective loss curve are $\mathcal{L}_{\text{total}} \approx 12.98$ at the first step, $\mathcal{L}_{\text{total}} \approx 1.42$ at 50,000 steps and $\mathcal{L}_{\text{total}} \approx 0.69$ at 300,000 steps



(a) Steep decline of the loss curve between 5,000 and 20,000 pre-training steps



(b) Less steep and more fluctuating decline of the loss curve between 150,000 (i.e., the checkpoint used for STonKGs_{150k}) and 165,000 pre-training steps



(c) Almost non-declining but highly fluctuating loss curve between 285,000 and 300,000 pre-training steps (i.e., the final checkpoint used for STonKGs_{300k})

Figure 21: Close-up snapshots of the pre-training loss curve of STonKGs over three chosen intervals. More specifically, the three figures show the loss curve of the shared pre-training procedure for STonKGs_{300k} and STonKGs_{150k} (i.e., an earlier checkpoint of STonKGs_{300k}), captured in three different intervals consisting of 15,000 training steps each. Again, the x-axis is showing the number of pre-training steps, whereas the y-axis is depicting the value of the loss. The loss is logged every 50 training steps, resulting in 300 data point for each of the three shown intervals. (Image source: own.)

shapes of both loss curves depicted in Figure 20 resemble typical logarithmic loss/cross-entropy loss curves. This is not surprising, given that the overall loss is calculated as the sum of three (or two) cross-entropy-based loss functions.

On a large scale (e.g., over the entire 150,000 or 300,000 training steps), both loss curves are seemingly smooth and hardly exhibit any irregularities. However, when taking a closer look (see Figure 21), it becomes evident that the amplitude of the fluctuations increasingly outweighs the decrease in loss over time. More precisely, this is depicted by the different levels of fluctuation in the depiction of the loss curve in equally sized intervals at three different stages of the pre-training procedure. In the first interval (starting at 5,000 training steps and ending at 20,000, see 21a), it can be seen that the overall difference of the loss between the beginning and the end of the interval ($\mathcal{L}_\alpha - \mathcal{L}_\omega \approx 3.8 - 2.2 = 1.6$) is greater than the fluctuations between neighboring logging steps. In the second interval (from 150,000 to 165,000 training steps, 21b), there is no such clear trend anymore. Instead, the amplitude of the fluctuation (lying at around 0.02 on average) is already in the same order of magnitude as the overall loss decrease in that interval (≈ 0.03). Lastly, in the third interval ranging from 285,000 to 300,000 (see Figure 21c), the fluctuations clearly show a greater impact on the change of the loss function compared to the overall decline.

Lastly, it should be highlighted that all reported values for the losses are based on the added partial losses from the individual pre-training objectives. Hence, it is not possible to conduct an in-depth analysis of the proportion of each of the three individual pre-training objectives (i.e., the MLM, MEM and NSP objectives for $\text{STonKGs}_{\text{150k}}$/$\text{STonKGs}_{\text{300k}}$ and the MLM and MEM objectives for $\text{STonKGs}_{\text{NO NSP}}$) on the overall loss at each logging step. However, indirect observations can be made about the role of the NSP training objective on the overall loss, as it is included in $\text{STonKGs}_{\text{150k}}$/$\text{STonKGs}_{\text{300k}}$ (see Figure 20a) but not in $\text{STonKGs}_{\text{NO NSP}}$ (see Figure 20b). Based on comparing the two respective loss curves, it becomes evident that the curves are almost identical in shape. This potentially indicates that the NSP objective is not harmful to the decrease of the loss during pre-training, and that it might only play a minor role in the overall progression of the loss. A further indication is given by the minor discrepancy between the two respective losses ($\mathcal{L}_{\text{total}}$ and $\mathcal{L}_{\text{NO NSP}}$) at 150,000 training steps. More specifically, while $\mathcal{L}_{\text{total}}$ is roughly equal to 0.91, $\mathcal{L}_{\text{NO NSP}}$ lies at around 0.76 (cf. Figure 20a and 20b), resulting in a difference of 0.15.

4.2 Fine-Tuning Benchmark Performances

Table 10 summarizes the mean and standard deviations for the weighted F1-scores recorded over five cross-validation folds for each proposed model evaluated on each constructed task. More specifically, the rows in Table 10 are listing all of the eight benchmark tasks, grouped and color-coded bases on their task types. The first five columns, on the other hand, are listing all five models, namely the NLP- and KG-baselines, as well as STonKGs_{150k}, STonKGs_{300k} and $\text{STonKGs}_{\text{NO NSP}}$. In each row, the model with the highest F1-score for the given task is highlighted in yellow. Moreover, the rightmost two columns are listing the absolute and relative differences (see Section 3.3.2) between the best baseline model and the best STonKGs variant on each task. Overall, the reported means of the F1-scores vary greatly depending on the concrete model and task, ranging from 0.020 to 0.995. Furthermore, the recorded standard deviations range from 0.325 to 0.000 (i.e., falling below the precision of the measurement consisting of three decimal places), but considerably low (i.e., lower than 0.01) in more than half (i.e., 22) of the 40 reported performances. Together with the low number of cross-validation folds, this results in a generally low explanatory power of the reported standard deviations. Therefore, the following comparisons are primarily supported by the mean F1-scores rather than the standard deviations.

The following subsections aim to provide three orthogonal perspectives for an in-depth analysis of the results reported in Table 10. First, Section 4.2.1 is taking a closer look at the differences between STonKGs and both baselines, as well as between the baseline models, to examine the initial hypothesis of this thesis (see Section 1.2). Then, the effects of the ablations are summarized in Section 4.2.2 to provide a closer understanding of how a decreased number of training steps and the exclusion of the NSP training objective affect the downstream benchmark performance of the STonKGs model. Lastly, Section 4.2.3 is reporting the differences in performance across the eight different tasks (rather than across the different models) and summarizes the main trends. It should be noted that absolute differences concerning the F1-scores are expressed in decimal numbers in the following. In contrast, percentages are indicating relative performance gains or losses (unless stated otherwise).

4.2.1 Differences Between STonKGs and the Baselines

As demonstrated by the task-specific best performances in Table 10, **STonKGs beats both baselines** in six out of eight classification tasks, namely **on the (2) interaction type, (3) cell line, (4) disease, (5) location, (7) annotation error (binary) and (8) annotation error (multiclass)** tasks. More precisely, all three STonKGs variants,

Model		NLP-Baseline	KG-Baseline	STonKGs 300k	STonKGs 150k	STonKGs no NSP	Absolute gain	Relative gain
Relation type	1) Polarity	**0.940 ± 0.001**	0.448 ± 0.119	0.930 ± 0.002	0.931 ± 0.002	0.918 ± 0.002	-0.009	-0.96%
	2) Interaction type	0.991 ± 0.001	0.945 ± 0.002	**0.995 ± 0.001**	**0.995 ± 0.000**	0.992 ± 0.000	+0.004	+0.40%
Context annotation	3) Cell line	0.238 ± 0.075	0.020 ± 0.007	0.252 ± 0.012	0.256 ± 0.015	**0.261 ± 0.022**	+0.023	+8.81%
	4) Disease	0.214 ± 0.011	0.030 ± 0.029	**0.248 ± 0.013**	0.240 ± 0.013	0.236 ± 0.008	+0.034	+15.89%
	5) Location	0.397 ± 0.022	0.295 ± 0.110	**0.405 ± 0.010**	0.404 ± 0.012	0.401 ± 0.011	+0.008	+2.02%
	6) Species	**0.865 ± 0.006**	0.670 ± 0.325	0.860 ± 0.005	0.860 ± 0.008	0.857 ± 0.007	-0.005	-0.58%
Annotation error	7) Binary	0.911 ± 0.006	0.708 ± 0.032	0.977 ± 0.003	**0.978 ± 0.002**	0.977 ± 0.002	+0.067	+7.35%
	8) Multi-class	0.881 ± 0.003	0.446 ± 0.079	**0.964 ± 0.005**	0.963 ± 0.005	0.960 ± 0.005	+0.083	+9.42%

Table 10: Reported mean and standard deviations of the weighted F1-scores recorded for all models on all eight tasks from the evaluation benchmark (i.e., the first five columns). A standard deviation of 0.000 does not necessarily indicate zero variance; it means that the value surpasses the precision in terms of recorded decimals (i.e., three decimal places). The two baselines (the NLP-baseline and the KG-baseline), as well as the three STonKGs variants (STonKGs$_{300k}$, STonKGs$_{150k}$ and STonKGs$_{NO\ NSP}$) are grouped together to facilitate the overall comparison. For each task, the best result is highlighted in yellow. Additionally, the absolute as well as the relative performance gains (see Section 3.3.2) between the best baseline and the best STonKGs variant are reported in the rightmost two columns and highlighted in green or red, depending on whether STonKGs improved over the baselines or not.

and not only the best STonKGs model, consistently results in higher F1-scores than the baselines on the named tasks. Moreover, the NLP-baseline persistently outperforms the KG-baseline on all eight tasks, with considerable margins ranging from 0.046 on task (2) to 0.492 on the (1) polarity task. Hence, the KG-baseline is the model with the lowest F1-scores on all tasks. For seven out of eight tasks, there is a gap of more than 0.100 (i.e., 10%[19]) between the KG-baseline and the next best model with regards to the respective F1-scores. Only for task (2), the KG-baseline scores roughly on par with the other models (discussed in more detail in Section 4.2.3). Additionally, the KG-baseline is the model for which the largest standard deviations are reported across all tasks (except for task (3)). However, since the initial hypothesis of this thesis is focusing on the aspect of an NLP x KGE Transformer beating both baselines, rather than on the comparison between the two baselines, the following paragraph is centered around the difference between the NLP-baseline (i.e., the best baseline model) and STonKGs (the NLP x KGE model). Additionally, it is important to note that while there are three STonKGs variants, the following explanations implicitly refer to the STonKGs model with the highest performance for a given task (since all STonKGs models consistently perform better or worse than the NLP-baseline on all tasks).

To compare the NLP-baseline to STonKGs in more depth, it makes sense to distinguish between *less complex* tasks (i.e., tasks with either two or three classes: (1) polarity, (2) interaction, (6) species and (7) annotation error (binary)) and *more complex* tasks (i.e., those with either five or ten classes: (3) cell line, (4) disease, (5) location and (8) annotation error (multiclass)). **For the more complex tasks, STonKGs clearly outperforms the NLP-baseline by a substantial margin**, specifically on task (3) (+0.023 absolute improvement), (4) (+0.034) and (7) (+0.083). Moreover, the magnitude of the differences becomes even more apparent when looking at the relative improvements (+8.81%, +15.89% and +9.42% for task (3), (4) and (7), respectively). Even for the fourth less complex task, namely for task (5), STonKGs still leads to a +2.02% relative performance gain. On the other hand, **for the less complex tasks, there is no clear trend indicating that STonKGs is outperforming the NLP-baseline** or the other way around. In more detail, while STonKGs is leading to a larger F1-score on task (2) (+0.004/+0.40%) and (7) (+0.067/+7.35%), this is not the case for task (1) and task (6), resulting in performance losses of -0.009 (-0.96%) and -0.005 (-0.58%), respectively. However, it should be highlighted that in three out of the four less complex tasks (task (1), (2) and (6)), the difference in F1-scores is smaller than 1%.

[19]This percentage is not to be confused with the relative performance gain.

4.2.2 Differences Between STonKGs and Its Ablations

Based on the reported performances in Table 10, one can recognize a clear trend: Across all tasks, the differences in performance between the three different STonKGs variants are consistently smaller than the differences between all STonKGs models and the baselines. For instance, on task (8), the performances of the STonKGs variants vary between 0.960 and 0.964 ($\Delta_{\text{within STonKGs}} = 0.004$), whereas the difference between the best baseline (i.e., the NLP-baseline with a score of 0.881) and the worst STonKGs model (i.e., the $\text{STonKGs}_{\text{NO NSP}}$ resulting in a F1-score of 0.960) is substantially larger ($\Delta_{\text{between models}} = 0.079 \approx 20 * \Delta_{\text{within STonKGs}}$). This suggests that the general incorporation of both the text and the KG modality has a greater effect on the performance than the change in pre-training steps or objectives (discussed in greater detail in Section 5). In general, out of the three constructed model variants in this thesis, **$\text{STonKGs}_{\text{300k}}$ results in the best overall performance** on the eight benchmark tasks, beating the other two variants on task (4), (5) and (8), and achieving the same F1-score as the next best model on task (2) and (6). On the other hand, $\text{STonKGs}_{\text{150k}}$ leads to the highest F1-scores on task (1) (still resulting in a lower score than the NLP-baseline) and task (7). Lastly, the $\text{STonKGs}_{\text{NO NSP}}$ model results in considerably lower performances than the other two variants, apart from task (3), in which it proves to be the best STonKGs variant.

However, on most tasks, the differences between the STonKGs variants are marginal, such as the +0.001 F1-score improvements on tasks (5) and (8). In fact, the reported standard deviations are greater than the difference across the means of the model variants in some cases (e.g., in task (5), the standard deviations of all three models are > 0.001, and thereby greater than the difference between $\text{STonKGs}_{\text{300k}}$ and $\text{STonKGs}_{\text{150k}}$). Moreover, it should be highlighted that the disparities between the STonKGs variants are not or just hardly passing the measurement precision of three decimal places. Only on task (3) and (4), there are apparent differences (≥ 0.004 and ≥ 0.008, respectively) between the F1-scores of the three STonKGs models.

4.2.3 Differences Across Tasks

Naturally, the most prominent variations across the F1-scores of the models on the eight tasks are linked to the nature of the respective classification settings, more specifically, the number of classes. For instance, in Table 10, one can observe that **the reported F1-scores on the three binary tasks** (i.e., task (1), (2) and (7)) **are considerably higher than the respective scores on the other tasks**, particularly for the NLP-baseline and the STonKGs variants. In the case of the two relation type tasks (task (1)

and (2)), another key distinctive feature is the increased dataset size compared to the other tasks. While there are $n = 78,979$ annotated text-triple pairs for the (1) polarity and (2) interaction tasks, the datasets of the other six tasks contain $n = 21,765$ text-triple pairs at most (i.e., less than 30% of the data used for the first two tasks, see Table 6).

Next, there are three tasks with three to eight classes: (5) location ($n_{\text{classes}} = 5$), (6) species ($n_{\text{classes}} = 3$) and (8) annotation error (multiclass) ($n_{\text{classes}} = 8$). Surprisingly, across these three tasks, the NLP-baseline and all STonKGs variants reached the highest F1-scores on the eight-class task (task (8)). Comparing these F1-scores to the performances on the three-class species task (task (6)) reveals another striking detail: While on task (6), the best STonKGs model is resulting in a F1-score of 0.860, STonKGs surpasses this performance by more than 0.1 (i.e., $\Delta_{\text{NLP (8)-(6)}} = 0.964 - 0.860 = 0.104$) on task (8). However, for the NLP-baseline, the difference across these two tasks is substantially smaller ($\Delta_{\text{STonKGs (8)-(6)}} = 0.881 - 0.865 = 0.016$). This may indicate that task (8) profits more from the joint use of both KG and text data than task (6) (which is further reflected by the NLP-baseline outperforming all STonKGs variants on that task). On the contrary, task (5) proved to be more challenging, resulting in a maximum F1-score of 0.405, which is a decrease of more than 0.4 compared to the performances on task (6) and (8).

Lastly, for the two ten-class tasks, namely the (3) cell line and (4) disease tasks, Table 10 clearly highlights the relatively **low performance of all models on the ten-class classification tasks** (i.e., F1-scores less or equal to 0.261) compared to the other six tasks. In addition to having the largest number of classes, these two tasks also contain the lowest number of text-triple pairs ($n = 3,760$ and $n = 4,586$ for task (3) and (4), respectively). These are also two of the tasks in which the best STonKGs variant led to the greatest relative performance gains compared to the NLP-baseline (see Section 4.2.1).

However, **the most notable discrepancies between the reported F1-scores across all eight tasks can be observed for the KG-baseline**, ranging from 0.020 on task (3) to 0.945 on task (2). In addition to the large deviation in the means of the F1-scores, the KG-baseline also demonstrates substantially larger variations in the reported standard deviations for the task-specific F1-scores compared to the NLP-baseline and the STonKGs variations. For instance, on task (2), the reported standard deviation is $\sigma_{\text{KG (2)}} = 0.002$, whereas on task (6), the reported value is $\sigma_{\text{KG (6)}} = 0.325$, which is more than 150 times larger (i.e., $\sigma_{\text{KG (6)}} = 162.5 * \sigma_{\text{KG (2)}}$). More specifically, there two tasks with very low standard deviations ($\sigma \leq 0.010$), task (2) and task (3) ($\sigma_{\text{KG (2)}} = 0.007$). Interestingly, these are the two tasks on which the KG-baseline performs best and worst, respectively.

Next, there are three tasks with medium-scale standard deviations ($0.010 < \sigma \leq 0.100$): Task (4), (7) and (8). While for task (7) and (8), the reported standard deviations are still roughly an order of magnitude smaller than the respective mean F1-scores, the standard deviation on task (4) ($\sigma_{\text{KG (4)}} = 0.029$) is almost as large as the mean F1-score itself ($\mu_{\text{KG (4)}} = 0.030$). Lastly, there are three remaining tasks with comparatively large standard deviations compared to the other tasks ($0.100 < \sigma$), namely task (1), (5) and (6). These large values represent large variations across the performances of the KG-baseline on varying train and test splits in the cross-validation procedure, which indicates the lack of robust model performance on changing dataset partitions. All in all, the substantial variations in the means and standard deviations of the reported F1-scores across tasks can potentially be linked to statistical biases in the training partitions of some fine-tuning datasets. In certain cases, this might motivate the KG-baseline to exploit correlations between certain nodes and specific labels present in the text-triple pairs rather than to learn meaningful feature representations.

To further examine this claim, it makes sense to investigate whether there are any statistical cues between the unique nodes present in each fine-tuning dataset and the labels for the respective text-triple pairs in which the nodes appear. For instance, the fine-tuning dataset used for the (2) interaction (as well as the (1) polarity) task contains 464 text-triple pairs, in which the node **p(HGNC:14931 ! SIRT3)** (i.e., the Sirtuin-3 (SIRT3) gene) is mentioned. In 462 cases, the respective text-triple pairs are associated with the *direct* label for task (2), compared to only two examples, which are labelled as *indirect*. For such nodes, **the KG-baseline is potentially able to exploit the correlation to the imbalanced class labels**, since in the test split, a text-triple pair containing a given node is likely to have the same label as another text-triple pair including the same node in the training split. In the provided example, the percentage of *direct* examples out of all text-triple pairs for the given node lies at $\frac{462}{464} \approx 99.6\%$. This imbalance for binary classification tasks (indifferent to which of the two labels is the dominating one) is referred to as the *class imbalance ratio* in the following.

Figure 22 depicts the results of such an analysis on all unique nodes of the dataset used for the (1) polarity and (2) interaction tasks. More specifically, on the x-axis, there are multiple groups of bars representing the set of unique nodes contained in at least two (or three, four or five) text-triple pairs in the dataset. The y-axis depicts the percentage of unique nodes in these groups with a class imbalance ratio of more than 90%. As shown by the large discrepancy between the bars belonging to the (1) polarity and the (2) interaction tasks in each group, there are substantially more nodes with a skewed

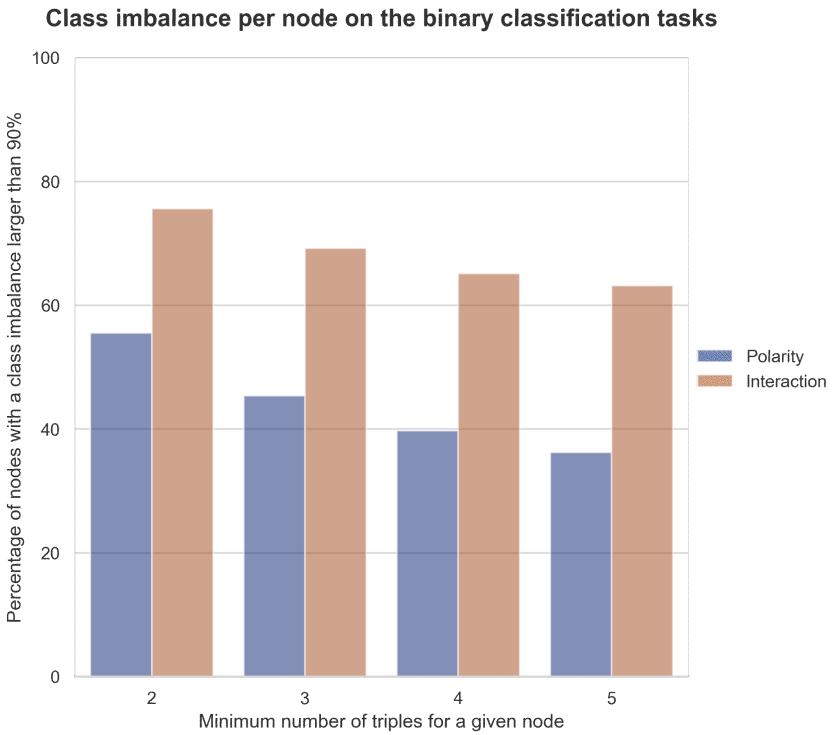


Figure 22: Percentage of unique nodes with a *class imbalance ratio* of greater than 90% for the (1) polarity and (2) interaction tasks (shown on the y-axis). The percentages are calculated for different groups containing all the unique nodes of the fine-tuning dataset used for task (1) and (2) that contain at least two, three, four or five text-triple pairs with a given node, respectively (i.e., the different groups on the x-axis). As shown by the large discrepancies between the bars belonging to the (1) polarity and (2) interaction tasks in each group, there are significantly more unique nodes with a large class imbalance ratio in task (2) than (1). (Image source: generated by Dr. Daniel Domingo Fernández)

class imbalance ratio for task (2) than task (1). This finding is particularly helpful for explaining the protruding performance of the KG-baseline on task (2) compared to all other tasks (discussed in more detail in the next section).

5 Discussion

Combining both text and KG data in a multimodal NLP x KGE Transformer-based ML model yields great potential for improving the downstream performance on a variety of classification tasks in a transfer learning setting. Such an NLP x KGE model can potentially learn meaningful links and interdependencies between sentence-level information present in unstructured text and subgraph-level information derived from structured KG data. Learning these links is based on a pre-training procedure on large quantities of text-triple pairs that encompass all biological knowledge available in biomedical literature. A three-step comparison was carried out to test the effectiveness of such a multimodal approach in this thesis. In more detail, the underlying hypothesis was that the proposed NLP x KGE model could improve downstream performances when compared to an NLP-baseline operating entirely on text data as well as a KG-baseline that solely uses KG data in a shared experimental setting (see Figure 1). The main results presented in Table 10 provide evidence in support of this initial hypothesis: The proposed NLP x KGE model, STonKGs, outperformed the two baselines on six out of a total of eight sequence classification tasks. Moreover, the pre-trained STonKGs models are made publicly available (see Section 3.4), which enables anyone to re-use the methodology developed in this thesis for their own use-cases dealing with the annotation (i.e., classification) of arbitrary text-triple pairs.

This section provides a closer examination of the central findings (reported in Section 4) and their implications. First, Section 5.1 is discussing further interpretations of the downstream model performances listed in Table 10 and the main trends summarized in Section 4.2. Next, Section 5.2 is covering the main limitations of the methodology applied in this thesis, followed by a summary of the major challenges encountered in this work in Section 5.3. Lastly, Section 5.4 is listing future research directions regarding the extension to other applications as well as further analyses that can provide a deeper understanding of this work.

5.1 Differences in Performances

Generally, STonKGs was able to outperform the other two baselines on six out of eight sequence classification tasks (as shown in Table 10). In particular, some of the largest relative performance gains were observed for the (3) cell line and (4) disease tasks, which are the tasks with both the largest number of classes as well as the lowest number of text-triple pairs. This finding suggests that the inclusion of both modalities in STonKGs is specifically beneficial for settings with a large number of classes and a low number

of examples for each class, which resembles potential real-world annotation settings the most (see Section 5.4). Surprisingly, there were not any substantial differences in the downstream performances of the three variations of the STonKGs model architecture (i.e., STonKGs$_{150k}$, STonKGs$_{300k}$ and STonKGs$_{NO\ NSP}$). This is possibly linked to i) the similar loss curves of STonKGs$_{150k}$/STonKGs$_{300k}$ and STonKGs$_{NO\ NSP}$ (cf. Figure 20a and 20b) as well as ii) the low proportion of the overall loss reduction that takes place between STonKGs$_{150k}$ and STonKGs$_{300k}$ (see Figure 20a). More specifically, towards the last portion of the 300,000 training steps, the levels of fluctuation clearly outweigh the overall decreasing trend of the loss curve (see Figure 21c), which likely indicates the minor effect of the additional pre-training steps in STonKGs$_{300k}$.

Overall, the KG-baseline model exhibited a comparatively poor performance across all eight benchmark tasks compared to the NLP-baseline and the three STonKGs variants. This is not surprising, given that the chosen classification tasks focus on predicting the context or relation type of a given text-triple pair, which is not explicitly captured by the KG embeddings learned by node2vec. Another reason that could potentially explain the relatively low performance is the employed fine-tuning strategy of the static KG-baseline model. Technically, the employed fine-tuning strategy is not strictly following the pre-training and fine-tuning setting of the transfer learning paradigm since the weights of the node2vec model are not adapted based on the task-specific cross-entropy losses in the fine-tuning procedures. Moreover, although the implemented fine-tuning architecture (consisting of the pooling procedure, the linear layer and the softmax activation function) aimed at providing a consistent comparison to the other two model architectures, it might not have been suitable for generating class predictions based on a set of static KG embeddings. For instance, the derived random walk-based embedding sequences do not contain the same special classification (i.e., `[CLS]`) token that is used in the other two Transformer-based models. Hence, a simple dimension-wise pooling procedure is expected to perform worse than using a dedicated aggregated `[CLS]` representation of an input sequence. After all, the order of magnitude of the effect from these complementary factors on the downstream performance of the KG-baseline cannot be precisely estimated, thus, requiring further experiments and analyses.

Interestingly, there are great differences across the task-specific performances for the KG-baseline. For instance, regarding the binary classification tasks, the model achieved an F1-score of 0.945 on the (2) interaction type task, compared to an F1-score of 0.448 on the (1) polarity task. Based on the demonstrated discrepancy in the class imbalance ratios between these two tasks (see Figure 22), there is evidence for substantial biases between

the unique nodes and their associated labels in each task. The KG-baseline possibly exploits these biases, which could explain the large difference in performance on these two tasks. More specifically, the biases could prevent the KG-baseline from learning appropriate values for the weights in the linear layer in the fine-tuning architecture component, which might also explain the large standard deviations of the model, particularly on the (1) polarity, (5) location and (6) species tasks. Moreover, the observed task-specific biases between the nodes and the class labels have further implications for STonKGs. More specifically, the same biases could be exploited in the attention mechanism (more precisely, through the learned attention coefficients) of STonKGs. Therefore, the reported results for both the KG-baseline and all STonKGs variants are potentially affected by these distortions. In the latter case, this considerably impacts the overall proof of concept of the multimodal Transformer, since it is hardly possible to tell apart whether the superior performance of STonKGs is caused by purposeful combinations of text and KG data or whether it is mainly influenced by potential biases present in the KG data[20]. However, the substantially lower standard deviations of all STonKGs variations indicate a more robust classification procedure compared to the KG-baseline.

Although such biases can exist in text data as well (e.g., certain words that are statistically correlated to specific class labels but not meaningfully related to a given fine-tuning task), the performance of the NLP-baseline proved to be both more robust (in terms of lower standard deviations) and higher compared to the KG-baseline. In fact, on many tasks, specifically on those with a comparably low number of classes (e.g., task (1), (2) and (6)), the NLP-baseline achieved a similar (if not better) performance compared to the best STonKGs variant. This indicates that for some tasks, the context information is sufficiently included in sentence-level text data rather than in subgraph-level KG data, meaning that a purely text-based Transformer can be adequate to solve these classification tasks. For instance, in the (6) species task, it is evident to assume that keywords such as *human, mouse* or *rat* mentioned in the text evidence (but not encoded anywhere in the random walk-based sequence data) are central indicators for the classification of a given text-triple pair. Furthermore, the lower standard deviations of both the NLP-baseline and STonKGs (compared to the KG-baseline) could indicate that fine-tuning all model weights (rather than only the fine-tuning components) is playing a central role in the robust adaptation of the pre-trained embedding representations.

[20] A careful investigation of the pre-trained attention coefficients of STonKGs on a small set of examples could potentially provide clarification. However, choosing the suitable Transformer layers, heads and examples easily leads to cherry-picking. Hence, this procedure is not ideal for explaining the general behavior of the model.

5.2 Limitations

While the main results of the thesis support the initial hypothesis, there are several limitations to the proposed methodology and the explanatory power of the reported results. First, as mentioned in the previous subsection, there are potential biases in the fine-tuning datasets used to construct the benchmark, which are likely to influence the overall expressiveness of the superiority of STonKGs on the benchmark. Hence, it is necessary to validate the three proposed model architectures on further carefully designed fine-tuning tasks that exclude such biases to provide stronger evidence regarding the central hypothesis. Moreover, although the pre-training and fine-tuning dataset splits were explicitly designed to be disjunct to prevent any information leakage (see Section 3.3.1), the fine-tuning datasets used in the benchmark are still based on INDRA statements, which are extracted using the same readers as the text-triple pairs employed in the pre-training procedure. As a result, possible inclinations of the readers employed by INDRA are potentially adopted in the creations of the fine-tuning datasets as well. For instance, most readers focus on high precision rather than high recall, which potentially favors the extraction of frequently mentioned and easily detectable biological processes rather than rare interactions.

To strengthen the evidence for a robust model performance of STonKGs, further text-triple pairs from sources and readers other than the ones used in INDRA need to be employed in the datasets. Regarding the fine-tuning datasets/tasks, the inclusion of more heterogeneous data sources is relatively straightforward, as it only requires extracting (or even manually curating) a comparatively small number (i.e., thousands) of labelled text-triple pairs from other sources. However, enriching the pre-training dataset with other sources is substantially harder since it requires a much larger number (i.e., millions) of examples. In the biomedical domain, there is a lack of publicly available large-scale datasets that can be used to create training examples in the required text-triple format. Essentially, this results in two options for obtaining large-scale text-triple datasets; either through collaborators who are willing to share their datasets or through unassisted data collection, which is expected to take a tremendous amount of time and effort[21]. Even though it was possible to leverage a considerably large number of raw statements through INDRA (i.e., over 35 million statements), both the applied filtering procedure (resulting in over 13 million triples, see Section 3.1.1) and the relatively short average token length (see Figure 14) cause the INDRA dataset to be smaller and less abundant in comparison to other sources from the general domain such as Wikidata.

[21] Hence, it would not have been feasible to collect an independent dataset suitable for both pre-training and fine-tuning within the scope of this thesis.

Next, there are additional limitations concerning numerous fixed (model) design choices, which could have been tested more thoroughly to provide further insights and possibly improve downstream model performances. For instance, fixed values were allocated for several hyperparameters of node2vec and the fine-tuning procedures for all three models, which left no possibility to explore further options that could have potentially improved the benchmark performances. More specifically, in node2vec, both q and p were set to one, meaning that neither local nor global exploration was favored when creating the random walks. Moreover, to guarantee the integration of KGEs in STonKGs, the random walk length l as well as the embedding dimension d had to be set to predetermined values (i.e., $l = 127$ and $d = 768$), which are still in the same order of magnitude, but considerably larger than the common choices for these hyperparameters. To be more precise, Grover and Leskovec tested a range of $[40, 100]$ for l in the original node2vec publication [GL16], and Mikolov et al. evaluated a range of $[50, 600]$ for the embedding dimension d in the word2vec model [Mik+13b] (that is internally used in node2vec). Therefore, a comparison between the node2vec model with the employed values in this thesis to another one with smaller values for l and d could have helped to estimate the effect of the unusual hyperparameter choices on the quality of the learned KGEs. For that, it would have been helpful to monitor the pre-training process of the node2vec model in more detail (e.g., the value of the log-likelihood used to optimize the internal word2vec model) to validate the effectiveness of the overall training procedure.

In addition, both the KG-baseline and STonKGs used a static lookup for the associated random walks for each node in the unique set of nodes in the INDRA pre-training dataset. Alternatively, it would have been possible to dynamically sample random walks for each node in the KG to generate the embedding sequences for each text-triple pair in the KG-baseline and STonKGs. However, that would have substantially increased the overall complexity of both models, which might have particularly led to an infeasible pre-training procedure for STonKGs. Another complementary strategy would be to leverage more sophisticated KGEMs such as GCNs or GATs to generate the embedding sequence for a given triple. This could also potentially eliminate one weakness of STonKGs, namely the static nature of the pre-trained node embeddings, which makes it impossible i) to derive an embedding representation for any node that is not included in the original pre-training partition of the INDRA dataset and ii) to incorporate relation types present in the triple data. In particular, the dependency on the set of nodes included in the pre-training procedure strongly hampers the application of STonKGs on data sources outside of INDRA. However, at least two aspects impede the use of such KGEMs in the proposed NLP x

KGE model architecture of this thesis. First, it is practically impossible to train embedding representations of millions of triples using convolutional or attention-based KGEMs (see Section 2.3). Secondly, to incorporate embeddings from a chosen KGE model into a multimodal Transformer, sequential representations (such as the random walks generated in node2vec) need to be created for a given triple first. Therefore, other KGEMs would require major adaptations prior to their integration into a multimodal Transformer.

Moreover, there are restraints in terms of the conclusions that can be drawn based on the proposed benchmark and evaluation procedure. All eight tasks are centered around sequence classification. Hence, all reported model performances must be interpreted with respect to this particular task type. To create a more all-encompassing benchmark, additional task types (with matching fine-tuning architecture modifications) must be incorporated as well (see Section 5.4). Next, although there is evidence that STonKGs is leading to improvements in performance, particularly on tasks with smaller datasets, profound conclusions can only be drawn based on a closer investigation of this aspect. For instance, it would have been possible to test the performances of all models on multiple smaller subsets of the same dataset for a given task (e.g., subsets with 1,000, 2,000, 5,000, and 10,000 entries). This could have possibly revealed more detailed insights about the potential benefits of STonKGs for transfer learning on small datasets.

Additionally, it can be argued that some of the fine-tuning tasks, specifically the binary ones, are too trivial to test for meaningful differences across the there proposed model architectures. This is particularly demonstrated in the (2) interaction task, in which four out of the five proposed model variants achieved an F1-score of > 0.99 (see Table 10). Combined with the lack of statistical testing based on the reported means and standard deviations (due to the low number of folds in the cross-validation procedure), one cannot argue using the statistical significance of the reported differences in model performances, which impedes the overall explanatory power. Furthermore, it can be argued that the binary tasks are unrelated to most real-world application scenarios, which are characterized by a large number of classes and a low number of examples per class. Lastly, all the comparisons made in this thesis are based on two custom-built baselines rather than on other biomedical LMs that incorporate KG data such as BioKGLM [Fei+20] or BERT-MK [He+20] (see Table 2). However, ensuring a fair comparison to STonKGs would not have been trivial since most of these models explicitly require entity linking (i.e., datasets that explicitly link single tokens to nodes, which is substantially different from the loose coupling of sentence-level text and subgraph-level KG data in this thesis).

5.3 Challenges Encountered in This Thesis

Although all of the previously outlined aims (see Section 1.3) have been achieved, there have been several challenges in the realization and implementation of the goals of this thesis. First, there were several complications concerning the employed INDRA dataset. The generation of the raw INDRA dataset was delayed by almost two months on the part of the INDRA team at HMS, leaving a smaller window than initially planned for the data preprocessing part and the execution of all pre-training procedures (including the ablations). In general, one major challenge was to find an appropriate balance between quality and quantity with regards to preprocessing the raw INDRA statements. As stated in Section 3.1.1, the main impediment has been the absence of text evidence in many statements. This did not leave much room for additional quality control mechanisms (for the remaining 13 million statements) since they would further reduce the number of text-triple pairs for pre-training and fine-tuning. Moreover, many aspects of the data dump generation were not entirely transparent (e.g., possible caveats concerning the readers, the proportion of statements coming from each reader or additional settings regarding the lengths of the extracted text evidences), making it hard to ensure the overall quality of the dataset. Furthermore, the annotations were much sparser than initially anticipated, which caused several problems for the creation of the fine-tuning tasks. The set of proposed fine-tuning tasks had to be changed multiple times, as some of the tasks did not have enough examples. Also, it would have been desirable to create tasks with a larger number of classes (e.g., $n_c = 20$ or even $n_c = 100$ classes). However, this was not possible due to the small number of text-triple pairs in the minority classes.

In addition, it is important to highlight the immense computational resources and amount of time that is required to pre-train STonKGs (see Section 3.4). Consequently, this meant that there was only a very limited number of attempts for pre-training the model and its variants on the preprocessed pre-training partition of INDRA. Hence, it was not possible to make changes to the preprocessed dataset or to the cross encoder of STonKGs later on, which implied that potential feedback could not be fully respected (especially given the overall time constraints of the thesis). Moreover, there were several issues regarding the implementation of the pre-training procedure. One of the central bottlenecks of this thesis was the runtime of the pre-training procedure. Thus, major efforts were invested into reducing the required runtime per training step (e.g., using FP16 precision instead of full precision, see Section 3.4). The `deepspeed` Python library [Ras+20] appeared to be a promising option for optimizing the runtime. However, it turned out that due to GPU driver incompatibilities, it was impossible to implement the library with the leveraged GPU resources (i.e., four NVIDIA A100s) in this thesis.

5.4 Future Work

There are plenty of future applications and developments that can be built around STonKGs. For instance, as literature grows, future versions of INDRA can be used to further pre-train STonKGs. Moreover, a public release of the full corpus of INDRA statements would be favorable so that other researchers can profit from the all-encompassing biomedical information contained in this large-scale dataset as well[22]. Similarly, other INDRA-equivalent KGs containing text-triple pairs could be leveraged, specifically to create fine-tuning tasks that can help to validate the robustness of STonKGs. Overall, the existing benchmark tasks based on INDRA and other potential tasks from external sources could be used to publicly release a benchmark. The research community can use this benchmark to evaluate new multimodal models that leverage both KG and text information in the biomedical domain. Additionally, further improvements to the existing STonKGs GitHub repository can be made. An increase in the overall user-friendliness may help to encourage the research community to use STonKGs on other use-cases. Furthermore, additional efforts can be made to update or replace the basic framework of the cross-encoder of STonKGs, given the recent advances in Transformer-based architectures beyond BERT. For instance, Clark et al. published the Generative Adversarial Network-like (GAN-like) "Efficiently Learning an Encoder that Classifies Token Replacements Accurately" (ELECTRA) model in 2020 [Cla+20]. In theory, it is possible to adapt ELECTRA to an unsupervised pre-training procedure on text-triple pairs, based on replacing some text tokens or nodes with plausible alternatives and training a discriminator on such examples. Similarly, another biomedical language model other than BioBERT could be employed as the NLP-backbone of STonKGs.

While the effectiveness of STonKGs has been demonstrated on various tasks in the proposed benchmark of this thesis, the model can be fine-tuned on numerous additional biomedical applications. For instance, based on fine-tuning STonKGs on non-INDRA triples that are annotated with labels from a fixed set of neurodegenerative diseases (e.g., coming from [Dom+17]), STonKGs can be used to classify to what neurodegenerative disease context a given text-triple pair belongs to (as indicated in [Bal+21]). As such, the described task is highly similar to the (4) disease task in the proposed benchmark. Still, it proves that the fine-tuning procedure of STonKGs is not limited to a particular type of biological context (i.e., the primarily cancer-related diseases in task (4)) but highly adaptive to the set of labels provided by the user of STonKGs. The extension of STonKGs to novel and carefully designed fine-tuning tasks on text-triple pairs from sources other

[22]The INDRA dataset is a proprietary dataset owned by the INDRA team at HMS, and it is planned to be publicly released in the near future.

than INDRA can also help in preventing node-specific biases present in at least one of the tasks included in the benchmark (as previously discussed in Section 5.1 and 5.2). In particular, additional tasks with a closer resemblance to the expected real-world use-cases (i.e., tasks with a larger number of classes) can be added to the benchmark to assess the potential of STonKGs to future application scenarios.

Moreover, such fined-tuned models could be deployed as automatic annotation systems that generate predictions for previously unseen text-triple pairs. Such predictions can assist human curators to enrich metadata in KGs, thereby significantly reducing costs and speeding up the curation process [Hoy+19]. Generally, as stated in Section 5.2, it is required that both nodes of a given text-triple pair are present in the INDRA KG. Future adaptations of STonKGs could be made to generate predictions in an inductive setting (i.e., making predictions with text-triple pairs containing nodes not present in INDRA). To do so, nodes can be inductively added to KG, and an inductive KGEM can be employed to generate the embeddings. Lastly, as indicated in Section 5.2, further efforts should be dedicated towards the extension of possible fine-tuning task types and respective fine-tuning model architecture adaptations. Such task types can include but are certainly not restricted to NER, RE, question answering, link prediction, and node classification.

6 Conclusion

This thesis has successfully demonstrated the effectiveness of the proposed NLP x KGE approach, STonKGs, in comparison to both the KG-baseline (based on node2vec) and the NLP-baseline (based on BioBERT) in a transfer learning setting. Based on pre-training the models on over 13 million text-triple pairs from INDRA in an unsupervised manner, the pre-trained embedding representations for each model were compared against each other on the constructed benchmark comprising eight sequence classification tasks. The eight fine-tuning tasks were grouped into three task types, representing different categories of biological applications. STonKGs achieved higher F1-scores than the other two baseline models on six out of eight tasks, and the largest performance gains were observed for tasks characterized by a small dataset and a large number of classes. This suggests that (semi-)automated annotation procedures can potentially profit from the inclusion of both text and KG data in STonKGs the most. However, further analysis and extensions of the benchmark to other data sources are required to make more profound conclusions. Lastly, through the public release of both the pre-trained STonKGs model and the code utilized in this thesis, it is possible to expand STonKGs to numerous future applications.

References

[Ahm+13] Amr Ahmed et al. "Distributed large-scale natural graph factorization". In: *Proceedings of the 22nd international conference on World Wide Web*. New York, NY, USA, May 2013, pp. 37–48. URL: https://doi.org/10.1145/2488388.2488393 (visited on 07/07/2021).

[Ala18a] Jay Alammar. *The Illustrated Transformer*. 2018. URL: http://jalammar.github.io/illustrated-transformer/ (visited on 06/05/2021).

[Ala18b] Jay Alammar. *Word2Vec - The Illustrated Word2vec*. 2018. URL: https://jalammar.github.io/illustrated-word2vec/ (visited on 06/05/2021).

[Ali+20] Mehdi Ali et al. "Bringing Light Into the Dark: A Large-scale Evaluation of Knowledge Graph Embedding Models Under a Unified Framework". In: *arXiv:2006.13365* (June 2020). URL: http://arxiv.org/abs/2006.13365 (visited on 08/07/2020).

[All+15] James Allen et al. "Complex Event Extraction using DRUM". In: *Proceedings of BioNLP 15*. 2015. URL: http://aclweb.org/anthology/W15-3801 (visited on 02/03/2021).

[Als+19] Emily Alsentzer et al. "Publicly Available Clinical BERT Embeddings". In: *Proceedings of the 2nd Clinical Natural Language Processing Workshop*. June 2019. URL: https://www.aclweb.org/anthology/W19-1909 (visited on 11/19/2020).

[Bag+18] Amir Ali Bagher Zadeh et al. "Multimodal Language Analysis in the Wild: CMU-MOSEI Dataset and Interpretable Dynamic Fusion Graph". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. July 2018. URL: https://aclanthology.org/P18-1208 (visited on 07/12/2021).

[Bah+15] Dzmitry Bahdanau et al. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *Proceedings of the 3rd International Conference on Learning Representations*. 2015. URL: http://arxiv.org/abs/1409.0473 (visited on 11/12/2020).

[Bal+21] Helena Balabin et al. "STonKGs: A Sophisticated Transformer Trained on Biomedical Text and Knowledge Graphs". In: *bioRxiv* (Aug. 2021). URL: https://www.biorxiv.org/content/10.1101/2021.08.17.456616v1 (visited on 08/20/2021).

[Bal21] Helena Balabin. "Exploratory Analysis of Natural Language Processing Approaches in Psychiatry and Clinical Psychology". Master's Project. Bonn Rhein-Sieg University of Applied Sciences, Feb. 2021.

[Ben+03] Yoshua Bengio et al. "A Neural Probabilistic Language Model". In: *Journal of Machine Learning Research* (Feb. 2003). URL: https://www.jmlr.org/papers/v3/bengio03a.html (visited on 12/30/2020).

[Bio21] National Center for Biotechnology Information. *PMC Overview*. 2021. URL: https://www.ncbi.nlm.nih.gov/pmc/about/intro/ (visited on 05/31/2021).

[Bir+20] Colin Birkenbihl et al. "Evaluating the Alzheimer's disease data landscape". In: *Alzheimer's & Dementia: Translational Research & Clinical Interventions* (2020). URL: https://alz-journals.onlinelibrary.wiley.com/doi/abs/10.1002/trc2.12102 (visited on 04/02/2021).

[BLC19] Iz Beltagy, Kyle Lo, and Arman Cohan. "SciBERT: A Pretrained Language Model for Scientific Text". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*. Nov. 2019. URL: https://www.aclweb.org/anthology/D19-1371 (visited on 11/19/2020).

[BN01] Mikhail Belkin and Partha Niyogi. "Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering". In: *Proceedings of the 14th International Conference on Neural Information Processing Systems*. 2001. URL: https://dl.acm.org/doi/10.5555/2980539.2980616 (visited on 07/07/2021).

[Bol+08] Kurt Bollacker et al. "Freebase: a collaboratively created graph database for structuring human knowledge". In: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. June 2008. URL: https://doi.org/10.1145/1376616.1376746 (visited on 04/05/2021).

[Bor+13] Antoine Bordes et al. "Translating embeddings for modeling multi-relational data". In: *Proceedings of the 26th International Conference on Neural Information Processing Systems*. Dec. 2013. URL: https://dl.acm.org/doi/10.5555/2999792.2999923 (visited on 11/26/2020).

[Bro+20] Tom Brown et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems 33*. 2020. URL: https://papers.nips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html (visited on 08/07/2020).

[Cel+15] Asli Celikyilmaz et al. “Enriching Word Embeddings Using Knowledge Graph for Semantic Tagging in Conversational Dialog Systems”. In: *AAAI Spring Symposia*. 2015. URL: https://www.aaai.org/ocs/index.php/SSS/SSS15/paper/viewFile/10333/10034 (visited on 04/05/2021).

[Cha+05] Nathalie Charnaux et al. “Syndecan-4 is a signaling molecule for stromal cell-derived factor-1 (SDF-1)/ CXCL12”. In: *The FEBS journal* (Apr. 2005). URL: https://febs.onlinelibrary.wiley.com/doi/10.1111/j.1742-4658.2005.04624.x (visited on 08/09/2021).

[Che+15] Xinlei Chen et al. “Microsoft COCO Captions: Data Collection and Evaluation Server”. In: *arXiv:1504.00325* (Apr. 2015). arXiv: 1504.00325. URL: http://arxiv.org/abs/1504.00325 (visited on 07/13/2021).

[Che+20a] Wenhu Chen et al. “KGPT: Knowledge-Grounded Pre-Training for Data-to-Text Generation”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Nov. 2020. URL: https://www.aclweb.org/anthology/2020.emnlp-main.697 (visited on 03/29/2021).

[Che+20b] Yen-Chun Chen et al. “UNITER: UNiversal Image-TExt Representation Learning”. In: *Proceedings of the 16th European Conference on Computer Vision*. Ed. by Andrea Vedaldi et al. 2020. URL: https://link.springer.com/chapter/10.1007/978-3-030-58577-8_7 (visited on 03/31/2021).

[Che+20c] Liying Cheng et al. “ENT-DESC: Entity Description Generation by Exploring Knowledge Graph”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Nov. 2020. URL: https://www.aclweb.org/anthology/2020.emnlp-main.90 (visited on 03/29/2021).

[Cla+20] Kevin Clark et al. “ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators”. In: *Eighth International Conference on Learning Representations*. Apr. 2020. URL: https://iclr.cc/virtual_2020/poster_r1xMH1BtvB.html (visited on 11/19/2020).

[Con19] Torch Contributors. *torch.nn — PyTorch 1.9.0 documentation*. 2019. URL: https://pytorch.org/docs/stable/nn.html#loss-functions (visited on 08/08/2021).

[Con21] HuggingFace Contributors. *Models - Hugging Face*. 2021. URL: https://huggingface.co/models (visited on 08/09/2021).

[Dav+17] Allan Peter Davis et al. "The Comparative Toxicogenomics Database: update 2017". In: *Nucleic Acids Research* (Jan. 2017). URL: https://doi.org/10.1093/nar/gkw838 (visited on 06/04/2021).

[Dea+98] Maria Deak et al. "Mitogen- and stress-activated protein kinase-1 (MSK1) is directly activated by MAPK and SAPK2/p38, and may mediate activation of CREB". In: *The EMBO journal* (Aug. 1998). URL: https://www.embopress.org/doi/full/10.1093/emboj/17.15.4426 (visited on 08/09/2021).

[Dem+10] Emek Demir et al. "The BioPAX community standard for pathway data sharing". In: *Nature Biotechnology* (Sept. 2010). URL: https://www.nature.com/articles/nbt.1666 (visited on 06/04/2021).

[Det+18] Tim Dettmers et al. "Convolutional 2D Knowledge Graph Embeddings". In: *Proceedings of the AAAI Conference on Artificial Intelligence 32*. Apr. 2018. URL: https://ojs.aaai.org/index.php/AAAI/article/view/11573 (visited on 07/07/2021).

[Dev+19] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. June 2019. URL: https://www.aclweb.org/anthology/N19-1423 (visited on 11/19/2020).

[Dev20] Scikit-Learn Developers. *API Reference — scikit-learn 0.24.2 documentation*. 2020. URL: https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics (visited on 08/08/2021).

[Dom+17] Daniel Domingo-Fernández et al. "Multimodal mechanistic signatures for neurodegenerative diseases (NeuroMMSig): a web server for mechanism enrichment". In: *Bioinformatics* (Nov. 2017). URL: https://doi.org/10.1093/bioinformatics/btx399 (visited on 08/20/2021).

[Dos+21] Alexey Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *arXiv:2010.11929* (June 2021). URL: http://arxiv.org/abs/2010.11929 (visited on 07/12/2021).

[DY19] Brati Das and Riqiang Yan. "A Close Look at BACE1 Inhibitors for Alzheimer's Disease Treatment". In: *CNS Drugs* (Mar. 2019). URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7330928/ (visited on 03/30/2021).

[Fab+16] Antonio Fabregat et al. "The Reactome pathway Knowledgebase". In: *Nucleic Acids Research* (Jan. 2016). URL: https://academic.oup.com/nar/article/46/D1/D649/4626770 (visited on 04/06/2021).

[Fei+20] Hao Fei et al. "Enriching contextualized language model from knowledge graph for biomedical information extraction". In: *Briefings in Bioinformatics* (June 2020). URL: https://academic.oup.com/bib/advance-article/doi/10.1093/bib/bbaa110/5854405 (visited on 02/09/2021).

[Fou19] The Wikimedia Foundation. *Wikidata*. 2019. URL: https://www.wikidata.org/wiki/Wikidata:Main_Page (visited on 04/06/2021).

[Fu+15] Gang Fu et al. "PubChemRDF: towards the semantic annotation of PubChem compound and substance databases". In: *Journal of Cheminformatics* (July 2015). URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4500850/ (visited on 08/02/2021).

[GB20] Benjamin Mate Gyori and John Ata Bachman. *Processors for knowledge input (indra.sources) — INDRA 1.19.0 documentation*. 2020. URL: https://indra.readthedocs.io/en/latest/modules/sources/ (visited on 02/25/2021).

[GB21] Benjamin Mate Gyori and John Ata Bachman. *indralab/indra_db*. May 2021. URL: https://github.com/indralab/indra_db (visited on 06/03/2021).

[GL16] Aditya Grover and Jure Leskovec. "node2vec: Scalable Feature Learning for Networks". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Aug. 2016. URL: https://doi.org/10.1145/2939672.2939754 (visited on 02/19/2021).

[Goo+16] Ian Goodfellow et al. *Deep Learning*. MIT Press, 2016. URL: https://www.deeplearningbook.org/ (visited on 11/05/2020).

[Goy+17] Yash Goyal et al. "Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering". In: *arXiv:1612.00837* (May 2017). URL: http://arxiv.org/abs/1612.00837 (visited on 07/13/2021).

[Gyo+17] Benjamin Mate Gyori et al. "From word models to executable models of signaling networks using automated assembly". In: *Molecular Systems Biology* (Nov. 2017). URL: https://www.embopress.org/doi/full/10.15252/msb.20177651 (visited on 02/24/2021).

[Has+13] Janna Hastings et al. “The ChEBI reference database and ontology for biologically relevant chemistry: enhancements for 2013”. In: *Nucleic Acids Research* (Jan. 2013). URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3531142/ (visited on 08/02/2021).

[He+20] Bin He et al. “BERT-MK: Integrating Graph Contextualized Knowledge into Pre-trained Language Models”. In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Nov. 2020. URL: https://www.aclweb.org/anthology/2020.findings-emnlp.207 (visited on 11/19/2020).

[He+21] Pengcheng He et al. “DeBERTa: Decoding-enhanced BERT with Disentangled Attention”. In: *arXiv:2006.03654* (Mar. 2021). URL: http://arxiv.org/abs/2006.03654 (visited on 06/04/2021).

[Hin90] Geoffrey Hinton. *UCI Machine Learning Repository: Kinship Data Set*. July 1990. URL: https://archive.ics.uci.edu/ml/datasets/kinship (visited on 06/27/2021).

[HKE18] Charles Tapley Hoyt, Andrej Konotopez, and Christian Ebeling. “PyBEL: a computational framework for Biological Expression Language”. In: *Bioinformatics* (Feb. 2018). URL: https://doi.org/10.1093/bioinformatics/btx660 (visited on 08/22/2021).

[Hoy+19] Charles Tapley Hoyt et al. “Re-curation and rational enrichment of knowledge graphs in Biological Expression Language”. In: *Database* (Jan. 2019). URL: https://doi.org/10.1093/database/baz068 (visited on 02/24/2021).

[HYL17] William Hamilton, Rex Ying, and Jure Leskovec. “Inductive representation learning on large graphs”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Dec. 2017. URL: https://dl.acm.org/doi/10.5555/3294771.3294869 (visited on 07/07/2021).

[HYL18] William Hamilton, Rex Ying, and Jure Leskovec. “Representation Learning on Graphs: Methods and Applications”. In: *arXiv:1709.05584* (Apr. 2018). URL: http://arxiv.org/abs/1709.05584 (visited on 06/10/2021).

[Ji+20] Yanrong Ji et al. “DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome”. In: *bioRxiv* (Sept. 2020). URL: https://www.biorxiv.org/content/10.1101/2020.09.17.301879v1 (visited on 07/12/2021).

[Ji+21] Shaoxiong Ji et al. "A Survey on Knowledge Graphs: Representation, Acquisition and Applications". In: *IEEE Transactions on Neural Networks and Learning Systems* (2021). arXiv: 2002.00388. URL: http://arxiv.org/abs/2002.00388 (visited on 06/23/2021).

[JM20] Daniel Jurafsky and James Martin. *Speech and Language Processing (Third Edition draft)*. Dec. 2020. URL: https://web.stanford.edu/~jurafsky/slp3/ (visited on 12/30/2020).

[Kam+21] Aishwarya Kamath et al. "MDETR - Modulated Detection for End-to-End Multi-Modal Understanding". In: *arXiv:2104.12763* (Apr. 2021). URL: http://arxiv.org/abs/2104.12763 (visited on 05/01/2021).

[Klu+16] Thomas Kluyver et al. "Jupyter Notebooks – a publishing format for reproducible computational workflows". In: *Proceedings of the 20th International Conference on Electronic Publishing*. Ed. by F. Loizides and B. Schmidt. 2016. URL: https://eprints.soton.ac.uk/403913/ (visited on 08/09/2021).

[Kow+19] Kamran Kowsari et al. "Text Classification Algorithms: A Survey". In: *Information* (Apr. 2019). URL: https://www.mdpi.com/2078-2489/10/4/150 (visited on 12/30/2020).

[KW17] Thomas Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *Proceedings of the 5th International Conference on Learning Representations*. 2017. URL: https://arxiv.org/abs/1609.02907 (visited on 07/07/2021).

[KW52] Jack Kiefer and Jacob Wolfowitz. "Stochastic estimation of the maximum of a regression function". In: *The Annals of Mathematical Statistics* (1952). URL: https://doi.org/10.1214/aoms/1177729392 (visited on 08/04/2021).

[Lee+20] Jinhyuk Lee et al. "BioBERT: a pre-trained biomedical language representation model for biomedical text mining". In: *Bioinformatics* (Feb. 2020). URL: https://academic.oup.com/bioinformatics/article/36/4/1234/5566506 (visited on 11/19/2020).

[Lew+20] Patrick Lewis et al. "Pretrained Language Models for Biomedical and Clinical Tasks: Understanding and Extending the State-of-the-Art". In: *Proceedings of the 3rd Clinical Natural Language Processing Workshop*. 2020. URL: https://www.aclweb.org/anthology/2020.clinicalnlp-1.17 (visited on 01/31/2021).

[LH19] Ilya Loshchilov and Frank Hutter. "Decoupled Weight Decay Regularization". In: *arXiv:1711.05101* (Jan. 2019). URL: http://arxiv.org/abs/1711.05101 (visited on 06/22/2021).

[LHZ21] Michelle Li, Kexin Huang, and Marinka Zitnik. "Representation Learning for Networks in Biology and Medicine: Advancements, Challenges, and Opportunities". In: *arXiv:2104.04883* (Apr. 2021). URL: http://arxiv.org/abs/2104.04883 (visited on 04/14/2021).

[Lip00] Carolyn Lipscomb. "Medical Subject Headings (MeSH)". In: *Bulletin of the Medical Library Association* (July 2000). URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC35238/ (visited on 08/09/2021).

[Liu+19] Yinhan Liu et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *arXiv:1907.11692* (July 2019). URL: http://arxiv.org/abs/1907.11692 (visited on 11/12/2020).

[Liu+20] Weijie Liu et al. "K-BERT: Enabling Language Representation with Knowledge Graph". In: *Proceedings of the 34th AAAI Conference on Artificial Intelligence*. Apr. 2020. URL: https://aaai.org/ojs/index.php/AAAI/article/view/5681 (visited on 04/01/2021).

[Lu+19] Jiasen Lu et al. "ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks". In: *Advances in Neural Information Processing Systems 32*. 2019. URL: https://papers.nips.cc/paper/2019/hash/c74d97b01eae257e44aa9d5bade97baf-Abstract.html (visited on 03/29/2021).

[Mar+09] Antonella Marucci et al. "The role of HSP70 on ENPP1 expression and insulin-receptor activation". In: *Journal of Molecular Medicine* (Feb. 2009). URL: https://www.ncbi.nlm.nih.gov/pmc/articles/pmc3760425/ (visited on 08/09/2021).

[Mar+20] Louis Martin et al. "CamemBERT: a Tasty French Language Model". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. July 2020. URL: https://www.aclweb.org/anthology/2020.acl-main.645 (visited on 11/19/2020).

[Med21a] National Library of Medicine. *About*. 2021. URL: https://pubmed.ncbi.nlm.nih.gov/about/ (visited on 05/31/2021).

[Med21b] National Library of Medicine. *MEDLINE Citation Counts by Year of Publication*. 2021. URL: https://www.nlm.nih.gov/bsd/medline_cit_counts_yr_pub.html (visited on 06/01/2021).

[Med21c] National Library of Medicine. *MEDLINE Overview*. 2021. URL: https://www.nlm.nih.gov/medline/medline_overview.html (visited on 05/31/2021).

[Mer21] Merriam-Webster. *Definition of CELL LINE*. 2021. URL: https://www.merriam-webster.com/dictionary/cell+line (visited on 08/02/2021).

[Mik+11] Tomas Mikolov et al. "Extensions of recurrent neural network language model". In: *Proceedings of the 2011 IEEE International Conference on Acoustics, Speech and Signal Processing*. May 2011. URL: https://ieeexplore.ieee.org/document/5947611 (visited on 12/30/2020).

[Mik+13a] Tomas Mikolov et al. "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems 26*. 2013. URL: https://arxiv.org/abs/1310.4546 (visited on 11/26/2020).

[Mik+13b] Tomas Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". In: *Proceedings of the 1st International Conference on Learning Representations*. Ed. by Yoshua Bengio and Yann LeCun. 2013. URL: http://arxiv.org/abs/1301.3781 (visited on 02/04/2021).

[Mio+18] Riccardo Miotto et al. "Deep learning for healthcare: review, opportunities and challenges". In: *Briefings in Bioinformatics* (Nov. 2018). URL: https://doi.org/10.1093/bib/bbx044 (visited on 04/02/2021).

[MLY17] Seonwoo Min, Byunghan Lee, and Sungroh Yoon. "Deep learning in bioinformatics". In: *Briefings in Bioinformatics* (Sept. 2017). URL: https://doi.org/10.1093/bib/bbw068 (visited on 04/02/2021).

[Mor+14] Daniel Morgan et al. "Mutation of putative GRK phosphorylation sites in the cannabinoid receptor 1 (CB1R) confers resistance to cannabinoid tolerance and hypersensitivity to cannabinoids in mice". In: *The Journal of Neuroscience* (Apr. 2014). URL: https://doi.org/10.1523/JNEUROSCI.3445-12.2014 (visited on 08/09/2021).

[NCB18] NCBI Resource Coordinators. "Database resources of the National Center for Biotechnology Information". In: *Nucleic Acids Research* (Jan. 2018). URL: https://academic.oup.com/nar/article/47/D1/D23/5160976 (visited on 06/02/2021).

[NRM15] Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. "Compositional Vector Space Models for Knowledge Base Completion". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. July 2015. URL: https://aclanthology.org/P15-1016 (visited on 07/07/2021).

[Ope21] OpenAI. *DALL·E: Creating Images from Text*. Jan. 2021. URL: https://openai.com/blog/dall-e/ (visited on 07/13/2021).

[Par+07] Min Jung Park et al. "Transcriptional repression of the gluconeogenic gene PEPCK by the orphan nuclear receptor SHP through inhibitory interaction with C/EBPalpha". In: *The Biochemical Journal* (Mar. 2007). URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1863575/ (visited on 08/09/2021).

[Pas+19] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. 2019. URL: https://arxiv.org/abs/1912.01703 (visited on 11/19/2020).

[Pen+17] Xiaoqing Peng et al. "Protein–protein interactions: detection, reliability assessment and applications". In: *Briefings in Bioinformatics* (Sept. 2017). URL: https://doi.org/10.1093/bib/bbw066 (visited on 08/08/2021).

[Per+16] Livia Perfetto et al. "SIGNOR: a database of causal relationships between biological entities". In: *Nucleic Acids Research* (Jan. 2016). URL: https://academic.oup.com/nar/article/44/D1/D548/2502592 (visited on 06/04/2021).

[Pet+18] Matthew Peters et al. "Deep Contextualized Word Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. June 2018. URL: https://www.aclweb.org/anthology/N18-1202 (visited on 11/26/2020).

[Pet+19] Matthew Peters et al. "Knowledge Enhanced Contextual Word Representations". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*. Nov. 2019. URL: https://www.aclweb.org/anthology/D19-1005 (visited on 11/19/2020).

[Pri21] The Trustees of Princeton University. *WordNet | A Lexical Database for English*. 2021. URL: https://wordnet.princeton.edu/ (visited on 04/06/2021).

[Rad+21] Alec Radford et al. "Learning Transferable Visual Models From Natural Language Supervision". In: *arXiv:2103.00020* (Feb. 2021). URL: http://arxiv.org/abs/2103.00020 (visited on 03/29/2021).

[Ram+21] Aditya Ramesh et al. "Zero-Shot Text-to-Image Generation". In: *arXiv:2102.12092* (Feb. 2021). URL: http://arxiv.org/abs/2102.12092 (visited on 03/29/2021).

[Ras+20] Jeff Rasley et al. "DeepSpeed: System Optimizations Enable Training Deep Learning Models with Over 100 Billion Parameters". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Aug. 2020. URL: https://doi.org/10.1145/3394486.3406703 (visited on 08/22/2021).

[RKR20] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. "A Primer in BERTology: What We Know About How BERT Works". In: *Transactions of the Association for Computational Linguistics* (Dec. 2020). URL: https://doi.org/10.1162/tacl_a_00349 (visited on 02/11/2021).

[Rol17] Jason Tyler Rolfe. "Discrete Variational Autoencoders". In: *Proceedings of the 5th International Conference on Learning Representations*. 2017. URL: https://arxiv.org/abs/1609.02200 (visited on 07/13/2021).

[Sar+14] Sirarat Sarntivijai et al. "CLO: The cell line ontology". In: *Journal of Biomedical Semantics* (Aug. 2014). URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4387853/ (visited on 08/09/2021).

[Sch+12] Lynn Marie Schriml et al. "Disease Ontology: a backbone for disease semantic integration". In: *Nucleic Acids Research* (Jan. 2012). URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3245088/ (visited on 01/17/2021).

[Sch+20a] Raphael Scheible et al. "GottBERT: a pure German Language Model". In: *arXiv:2012.02110* (Dec. 2020). URL: http://arxiv.org/abs/2012.02110 (visited on 06/18/2021).

[Sch+20b] Conrad Schoch et al. "NCBI Taxonomy: a comprehensive update on curation, resources and tools". In: *Database* (Jan. 2020). URL: https://doi.org/10.1093/database/baaa062 (visited on 08/08/2021).

[SG18] Anastasia Shimorina and Claire Gardent. “Handling Rare Items in Data-to-Text Generation”. In: *Proceedings of the 11th International Conference on Natural Language Generation*. Nov. 2018. URL: https://aclanthology.org/W18-6543 (visited on 07/13/2021).

[Sha+19] Rebecca Sharp et al. “Eidos, INDRA, & Delphi: From Free Text to Executable Causal Models”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*. June 2019. URL: https://www.aclweb.org/anthology/N19-4008 (visited on 02/24/2021).

[Shi+20] Jun Shi et al. “Challenges of drug development during the COVID-19 pandemic: Key considerations for clinical trial designs”. In: *British Journal of Clinical Pharmacology* (Oct. 2020). URL: https://bpspubs.onlinelibrary.wiley.com/doi/abs/10.1111/bcp.14629 (visited on 03/30/2021).

[Sla14] Ted Slater. “Recent advances in modeling languages for pathway maps and computable biological networks”. In: *Drug Discovery Today* (Feb. 2014). URL: https://www.sciencedirect.com/science/article/pii/S1359644614000063 (visited on 06/04/2021).

[SN12] Mike Schuster and Kaisuke Nakajima. “Japanese and Korean voice search”. In: *Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing*. Mar. 2012. URL: https://ieeexplore.ieee.org/document/6289079 (visited on 06/05/2021).

[Su+20] Chang Su et al. “Network embedding in biomedical data science”. In: *Briefings in Bioinformatics* (Jan. 2020). URL: https://doi.org/10.1093/bib/bby117 (visited on 04/02/2021).

[Sun+19] Yu Sun et al. “ERNIE: Enhanced Representation through Knowledge Integration”. In: *arXiv:1904.09223* (Apr. 2019). URL: http://arxiv.org/abs/1904.09223 (visited on 08/03/2020).

[Sun+20] Tianxiang Sun et al. “CoLAKE: Contextualized Language and Knowledge Embedding”. In: *Proceedings of the 28th International Conference on Computational Linguistics*. Dec. 2020. URL: https://aclanthology.org/2020.coling-main.327 (visited on 07/07/2021).

[Tan+20] Mohammad Tanveer et al. “Machine Learning Techniques for the Diagnosis of Alzheimer’s Disease: A Review”. In: *ACM Transactions on Multimedia Computing, Communications and Applications* (Apr. 2020). URL: https://dl.acm.org/doi/10.1145/3344998 (visited on 04/06/2021).

[Tay53] Wilson Taylor. "Cloze Procedure: A New Tool for Measuring Readability". In: *Journalism Bulletin* (Sept. 1953). URL: https://journals.sagepub.com/doi/10.1177/107769905303000401 (visited on 11/26/2020).

[TB19] Hao Tan and Mohit Bansal. "LXMERT: Learning Cross-Modality Encoder Representations from Transformers". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*. Nov. 2019. URL: https://www.aclweb.org/anthology/D19-1514 (visited on 03/26/2021).

[The15] The UniProt Consortium. "UniProt: a hub for protein information". In: *Nucleic Acids Research* (Jan. 2015). URL: https://doi.org/10.1093/nar/gku989 (visited on 04/06/2021).

[Tro+04] Sonya Trombino et al. "Alpha7-nicotinic acetylcholine receptors affect growth regulation of human mesothelioma cells". In: *Cancer Research* (Jan. 2004). URL: https://cancerres.aacrjournals.org/content/64/1/135.long (visited on 08/09/2021).

[Tsa+19] Yao-Hung Hubert Tsai et al. "Multimodal Transformer for Unaligned Multimodal Language Sequences". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. July 2019. URL: https://www.aclweb.org/anthology/P19-1656 (visited on 02/09/2021).

[Twe+21] Susan Tweedie et al. "Genenames.org: the HGNC and VGNC resources in 2021". In: *Nucleic Acids Research* (Jan. 2021). URL: https://doi.org/10.1093/nar/gkaa980 (visited on 08/02/2021).

[Val+18] Marco Valenzuela-Escárcega et al. "Large-scale automated machine reading discovers new cancer-driving mechanisms". In: *Database* (Jan. 2018). URL: https://doi.org/10.1093/database/bay098 (visited on 02/25/2021).

[Vas+17] Ashish Vaswani et al. "Attention Is All You Need". In: *Proceedings of the International Conference on Neural Information Processing Systems 31*. Dec. 2017. URL: https://arxiv.org/abs/1706.03762 (visited on 11/19/2020).

[VD95] Guido Van Rossum and Fred Drake. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995. URL: https://ir.cwi.nl/pub/5008 (visited on 01/29/2021).

[Vel+18] Petar Veličković et al. “Graph Attention Networks”. In: *arXiv:1710.10903* (Feb. 2018). URL: http://arxiv.org/abs/1710.10903 (visited on 08/18/2020).

[Vri+19] Wietse de Vries et al. “BERTje: A Dutch BERT Model”. In: *arXiv:1912.09582* (Dec. 2019). URL: http://arxiv.org/abs/1912.09582 (visited on 11/12/2020).

[Wan+14] Zhen Wang et al. “Knowledge Graph and Text Jointly Embedding”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Oct. 2014. URL: https://www.aclweb.org/anthology/D14-1167 (visited on 03/29/2021).

[Wan+18] Alex Wang et al. “GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding”. In: *Proceedings of the 2018 EMNLP BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*. 2018. URL: http://aclweb.org/anthology/W18-5446 (visited on 12/13/2020).

[Wan+19a] Alex Wang et al. “SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems”. In: *Advances in Neural Information Processing Systems 31*. 2019. URL: https://arxiv.org/abs/1905.00537 (visited on 11/12/2020).

[Wan+19b] Xiaozhi Wang et al. “KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation”. In: *arXiv:1911.06136* (Nov. 2019). URL: http://arxiv.org/abs/1911.06136 (visited on 02/09/2021).

[Wan+20] Ruize Wang et al. “K-Adapter: Infusing Knowledge into Pre-Trained Models with Adapters”. In: *arXiv:2002.01808* (Feb. 2020). URL: http://arxiv.org/abs/2002.01808 (visited on 02/09/2021).

[Wis+18] David Wishart et al. “DrugBank 5.0: a major update to the DrugBank database for 2018”. In: *Nucleic Acids Research* (Jan. 2018). URL: https://academic.oup.com/nar/article/46/D1/D1074/4602867 (visited on 06/04/2021).

[Wol+20] Thomas Wolf et al. “Transformers: State-of-the-Art Natural Language Processing”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. July 2020. URL: https://aclanthology.org/2020.emnlp-demos.6/ (visited on 08/07/2020).

[Wu+16] Yonghui Wu et al. "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation". In: *arXiv:1609.08144* (Oct. 2016). URL: http://arxiv.org/abs/1609.08144 (visited on 11/12/2020).

[Xia+17] Han Xiao et al. "SSP: Semantic Space Projection for Knowledge Graph Embedding with Text Descriptions". In: *Proceedings of the AAAI Conference on Artificial Intelligence 31* (Feb. 2017). URL: https://arxiv.org/abs/1604.04835 (visited on 07/08/2021).

[Xie+16] Ruobing Xie et al. "Representation learning of knowledge graphs with entity descriptions". In: *Proceedings of the 30th AAAI Conference on Artificial Intelligence*. Feb. 2016. URL: https://dl.acm.org/doi/10.5555/3016100.3016273 (visited on 07/08/2021).

[Zha+19] Zhengyan Zhang et al. "ERNIE: Enhanced Language Representation with Informative Entities". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. May 2019. URL: https://www.aclweb.org/anthology/P19-1139 (visited on 11/19/2020).

[Zhe+18] Wei Zheng et al. "A document level neural model integrated domain knowledge for chemical-induced disease relations". In: *BMC Bioinformatics* (Sept. 2018). URL: https://doi.org/10.1186/s12859-018-2316-x (visited on 03/31/2021).

[Zho+18] Huiwei Zhou et al. "Chemical-induced disease relation extraction with dependency information and prior knowledge". In: *Journal of Biomedical Informatics* (Aug. 2018). URL: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-019-2884-4 (visited on 03/31/2021).

[Zho+19] Huiwei Zhou et al. "Knowledge-guided convolutional networks for chemical-disease relation extraction". In: *BMC Bioinformatics* (May 2019). URL: https://doi.org/10.1186/s12859-019-2873-7 (visited on 03/31/2021).

[Zhu+15] Yukun Zhu et al. "Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books". In: *Proceedings of the 2015 IEEE International Conference on Computer Vision*. 2015. URL: https://arxiv.org/abs/1506.06724 (visited on 01/08/2021).